INFORME DE GUÍA PRÁCTICA

1 Portada

Tema:	APE 7. Uso de Arreglos Unidireccionales
Unidad de Organización Curricular:	BASICA.
Nivel y Paralelo:	PRIMERO - B
Alumnos participantes:	Apellidos y Rovalino Ortega Mateo Rafael
	.....
Asignatura:	FUNDAMENTOS DE PROGRAMACION
Docente:	Ing. Hernan Naranjo, Mg.

2 Informe de guía práctica

2.1 Objetivos

2.1.1 General

Desarrollar métodos para aplicar la reutilización de código y simplificar procesos .

2.1.2 Específicos

- Reforzar las sentencias condicionales y cíclicas dadas en clase por el docente mediante la auto evaluación.
- Generar lógica de programación mediante la resolución de cada uno de los ejercicios propuestos en la guía de trabajo.
- Mejorar nuestra cognición para poder solucionar problemas mas compuestos en la asignatura.

2.2 Modalidad

Presencial

2.3 Tiempo de duración

Presenciales: 2
No presenciales: 0

2.4 Instrucciones

Acciones previas: Ingrese al aula virtual de la asignatura en donde se halla el trabajo del tema tratado. Elabore el trabajo siguiendo las instrucciones solicitadas.

2.5 Listado de equipos, materiales y recursos

Listado de equipos y materiales generales empleados en la guía práctica:

-

TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

☐ Plataformas educativas

☐ Simuladores y laboratorios virtuales

☐ Aplicaciones educativas

☐ Recursos audiovisuales

☐ Gamificación

☐ Inteligencia Artificial

- Otros (Especifique): _____

2.6 Actividades por desarrollar

Con la información propuesta en la tarea: Analice la información de cada uno de los ejercicios propuestos Realice el Diseño, prueba, codificación y posteriormente compilar, ejecutar y verificar el resultado. Arribe a conclusiones. Subir a la plataforma el archivo en formato .PDF del trabajo hasta la fecha indicada.

2.7 Resultados obtenidos

Ejercicios :

Ejercicio 1: Crear un vector de números enteros y mostrar su contenido por pantalla.

Para implementar la suma de dos vectores en Java usando arrays, definí un método llamado sumarVectores que recibe dos vectores representados como arreglos de tipo double[], cada uno con dos elementos que representan las componentes x e y del vector.

Dentro del método, creé un nuevo arreglo resultado de tamaño 2 para almacenar las componentes de la suma vectorial. La suma se realiza componente a componente, es decir:

La componente x del resultado es la suma de las componentes x de los vectores de entrada (v1[0] + v2[0]).

La componente y del resultado es la suma de las componentes y de los vectores de entrada (v1[1] + v2[1]).

El método devuelve este nuevo vector resultado.

En el método main, inicialicé dos vectores como arreglos double[] con valores específicos, luego llamé a la función sumarVectores y almacené el resultado en otro arreglo. Finalmente, imprimí el resultado formateado para mostrar las componentes x e y de la suma.

Ejercicio 2: Calcular la suma de los elementos de un vector de enteros.

Para calcular la magnitud (o longitud) de un vector en Java usando arrays, definí un método llamado magnitud que recibe un vector representado como un arreglo de tipo double[], con dos elementos que corresponden a las componentes x e y del vector.

Dentro del método, apliqué la fórmula matemática para la magnitud de un vector bidimensional: la raíz cuadrada de la suma de los cuadrados de sus componentes. Es decir, calculé ($x^2 + y^2$) usando Math.sqrt para la raíz cuadrada y operaciones de multiplicación para los cuadrados (vector[0] * vector[0] y vector[1] * vector[1]).

El método devuelve el valor de la magnitud como un número de tipo double.

En el método main, definí un vector como un arreglo double[] con valores específicos para sus componentes, llamé a la función magnitud con este vector y mostré el resultado por consola.

Ejercicio 3: Encontrar el valor máximo en un vector de números reales.

Para calcular el producto escalar de dos vectores en Java usando arrays, creé un método llamado productoEscalar que recibe dos vectores representados como arreglos de tipo double[], cada uno con dos elementos que representan las componentes x e y.

Dentro del método, implementé la fórmula del producto escalar en 2D: se multiplica la componente x del primer vector por la componente x del segundo vector, y se suma al producto de las componentes y de ambos vectores. Es decir, v1[0] * v2[0] + v1[1] * v2[1].

El método devuelve el resultado como un valor double.

En el método main, definí dos vectores como arreglos double[] con valores dados para cada componente, llamé al método productoEscalar pasando ambos vectores, y finalmente imprimí el resultado en la consola.

Ejercicio 4: Invertir el contenido de un vector.

Este programa en Java calcula el ángulo entre dos vectores en dos dimensiones representados como arreglos de tipo double[]. Para lograrlo, definí tres métodos principales:

productoEscalar: Recibe dos vectores y calcula su producto escalar, que es la suma de los productos de sus componentes correspondientes. Matemáticamente, para vectores

Primero calcula el producto escalar de los dos vectores, luego las magnitudes individuales y multiplica estas. Después calcula el arcocoseno del cociente y convierte el resultado de radianes a grados.

En el método main, definí dos vectores ortogonales (perpendiculares) como arreglos, llamé al método anguloEntreVectores para calcular el ángulo entre ellos y finalmente imprimí el resultado en consola.

Ejercicio 5: Verificar si un número específico se encuentra dentro de un vector.

Este programa en Java descompone un vector dado en su forma polar (magnitud y ángulo en grados) a sus componentes cartesianas (coordenadas x y y) usando arreglos double[].

Definí el método descomponerVector que recibe dos parámetros: la magnitud del vector y su ángulo en grados.

Dentro del método, convertí el ángulo de grados a radianes usando Math.toRadians porque las funciones trigonométricas de Java (Math.cos y Math.sin) trabajan con radianes.

En el método main, asigné valores para la magnitud y el ángulo, llamé a descomponerVector y mostré el vector resultante en forma de coordenadas cartesianas.

Ejercicio 6: Contar cuántas veces aparece un número dado en un vector.

Arreglos double[] para representar puntos en el plano con coordenadas x y y.

Funciones matemáticas de Java:

Math.pow para elevar al cuadrado las diferencias de coordenadas.

Math.sqrt para calcular la raíz cuadrada de la suma de los cuadrados.

Operaciones aritméticas para aplicar la fórmula de la distancia.

Retorno de un valor double con el resultado.

Impresión en consola para mostrar la distancia calculada.

Ejercicio 7: Ordenar un vector de enteros de menor a mayor.

Este programa en Java determina si dos vectores en 2D son ortogonales (perpendiculares) utilizando arreglos double[] para representar los vectores. Arreglos double[] para representar vectores en 2D.

La operación matemática del producto escalar para vectores.

Comparación simple con == 0 para verificar ortogonalidad.

Retorno de un valor booleano para indicar si los vectores son ortogonales o no.

Impresión en consola para mostrar el resultado. Ejercicio 8: Comparar dos vectores y determinar si son iguales.

Este programa en Java rota un vector bidimensional usando un arreglo double[] para representar el vector.

Definí el método rotar, que recibe dos parámetros: un vector v representado como un arreglo double[] con dos elementos (las componentes x y y) y un ángulo en grados grados que indica cuánto se debe rotar el vector.

Dentro de rotar:

Convertí los grados a radianes usando Math.toRadians(grados), ya que las funciones trigonométricas en Java usan radianes.

Ejercicio 9: Rotar un vector una posición a la derecha.

Definí el método normalizar, que recibe un vector v, calcula su magnitud llamando al método magnitud y luego crea un nuevo vector donde cada componente se divide entre la magnitud para obtener un vector de longitud 1 (unitario). Las componentes normalizadas son:

Llamé al método normalizar para obtener el vector unitario correspondiente, y almacené el resultado en unitario.

Imprimí el vector unitario mostrando sus componentes.

Qué utilicé:

Arreglos double[] para representar vectores bidimensionales.

La función Math.sqrt para calcular la raíz cuadrada al obtener la magnitud.

Operaciones aritméticas básicas para calcular la magnitud y normalizar.

Creación y retorno de un nuevo arreglo para el vector normalizado.

Impresión en consola para mostrar el resultado.

Ejercicio 10: Multiplicar todos los elementos de un vector por un escalar.

Este programa en Java convierte un vector definido en coordenadas polares (magnitud y ángulo) a su representación en coordenadas cartesianas (x, y) usando arreglos double[].

Definí el método polarACartesiano que recibe dos parámetros:

magnitud: la longitud del vector.

angulo: el ángulo en grados que forma el vector con el eje x.

Convertí el ángulo de grados a radianes usando Math.toRadians(angulo) porque las funciones trigonométricas en Java (Math.cos y Math.sin) trabajan con radianes.

Ejercicio 11: Calcular la proyección de un vector sobre otro.

Arreglos double[] para representar vectores 2D con sus componentes x y y.

Producto escalar para calcular la relación entre dos vectores.

Operaciones aritméticas básicas para cálculo de factores y componentes.

Funciones que devuelven nuevos arreglos para mantener la inmutabilidad de los vectores originales.

Impresión en consola para mostrar el resultado.

Ejercicio 12: Realizar una interpolación lineal entre dos vectores.

Este programa en Java realiza la interpolación lineal entre dos vectores en 2D representados como arreglos de tipo double[].

Definí el método interpolar que recibe dos vectores a y b (cada uno con dos componentes: x y y) y un parámetro

t que indica el porcentaje de interpolación entre estos vectores. Arreglos double[] para representar vectores 2D.

La operación matemática de interpolación lineal para obtener un vector entre dos vectores dados.

Cálculos con operaciones aritméticas básicas para interpolar cada componente.

Método que retorna un nuevo arreglo para representar el vector interpolado.

Impresión por consola para mostrar el resultado final.

Ejercicio 13: Reflejar un vector respecto al eje X.

En este programa en Java, implementé la función de interpolación lineal entre dos vectores en dos dimensiones representados como arreglos double[].

Definí un método llamado interpolar que toma tres parámetros: dos vectores a y b (cada uno con dos componentes: x y y) y un valor t que representa la fracción o el parámetro de interpolación, normalmente entre 0 y 1.

Arreglos double[] para representar vectores en 2D.

Operaciones matemáticas básicas (suma, resta, multiplicación) para calcular la interpolación lineal.

Método que devuelve un nuevo arreglo con el resultado.

Función System.out.println para mostrar el resultado en pantalla.

La fórmula de interpolación lineal que se usa comúnmente en geometría y gráficos por computadora para calcular puntos entre dos posiciones.

Ejercicio 14: Obtener un vector perpendicular a un vector dado.

En este programa de Java, creé una función para calcular el vector perpendicular de un vector en 2 dimensiones representado mediante un arreglo double[].

Definí un método estático llamado perpendicular que recibe como parámetro un arreglo double[] v, donde v[0] representa la componente

La función retorna un nuevo arreglo con las componentes del vector perpendicular calculadas como -v[1], v[0].

Utilicé un arreglo de tipo double[] para representar vectores en 2D, facilitando la manipulación y acceso a las componentes.

La fórmula matemática para obtener un vector perpendicular en dos dimensiones, que consiste en intercambiar las componentes x y y cambiar el signo de una de ellas.

Creación de un nuevo arreglo para retornar el vector perpendicular calculado.

El método System.out.println para mostrar el resultado en la consola.

Ejercicio 15: Determinar si tres puntos forman un triángulo rectángulo usando sus coordenadas.

En este programa de Java, desarrollé una función para determinar si un triángulo definido por tres puntos en el plano cartesiano es un triángulo rectángulo.

Uso la propiedad del teorema de Pitágoras que establece que en un triángulo rectángulo, la suma de los cuadrados de los catetos es igual al cuadrado de la hipotenusa. Para verificar esto comparo las tres combinaciones posibles:

Para evitar errores numéricos por precisión limitada de los números de punto flotante, uso una tolerancia muy pequeña (1e-9) al comparar si la suma es igual al tercer lado.

Retorno true si alguna de las condiciones se cumple, indicando que el triángulo es rectángulo; de lo contrario, retorno false.

El teorema de Pitágoras para validar si un triángulo es rectángulo.

El método Math.pow para elevar al cuadrado las diferencias de coordenadas.

Comparación con tolerancia (1e-9) para evitar errores por precisión de punto flotante.

El método System.out.println para mostrar el resultado en consola.

Ejercicio 16: Calcular la suma acumulada de un arreglo de vectores.

En este programa de Java, desarrollé una función para sumar vectores bidimensionales y un proceso para acumular la suma de varios vectores almacenados en un arreglo.

Definí un método llamado suma que recibe dos vectores representados como arreglos double[] con dos elementos cada uno, correspondientes a las componentes

Dentro del método suma, calculo la suma componente a componente, es decir, sumo las componentes x de ambos vectores y las componentes y de ambos vectores por separado. El resultado es un nuevo vector con esas componentes sumadas.

En el método main, creé un arreglo bidimensional double[][] vectores que contiene tres vectores, cada uno con dos componentes.

Inicialicé un vector acumulador suma en [0,0][0,0] para almacenar la suma progresiva.

Utilicé un ciclo for-each para recorrer cada vector en el arreglo vectores y actualizar el vector acumulador sumándole el vector actual usando el método suma.

Finalmente, imprimí la suma acumulada de todos los vectores en la consola.

Ejercicio 17: Determinar si dos vectores son ortogonales.

En este programa en Java, implementé una función para calcular el producto punto (o producto escalar) entre dos vectores bidimensionales, y luego utilicé esa función para determinar si dos vectores son ortogonales. Arreglos unidimensionales double[] para representar vectores 2D con dos componentes.

Operación matemática del producto escalar (producto punto) para vectores.

Comparación simple para verificar si el producto escalar es cero (lo que indica ortogonalidad).

Métodos estáticos para organizar el código y facilitar su reutilización.

Estructura básica de Java con método main para ejecutar el programa.

System.out.println para mostrar el resultado en consola.

Ejercicio 18: Invertir (cambiar la dirección) de un vector.

En este programa escribí una función que invierte un vector en 2D representado por un arreglo de tipo double. Invertir un vector significa cambiar el signo de todas sus componentes. Para ello:

Creé un método estático llamado invertir, que recibe como parámetro un vector v representado por un arreglo double[] de dos elementos.

Arreglos (double[]) para representar los vectores en dos dimensiones.

Un método estático (invertir) para realizar la inversión del vector.

Operaciones aritméticas simples: en este caso, cambiar el signo de cada componente multiplicando por -1.

Impresión por consola con System.out.println para mostrar el resultado.

La estructura básica de Java, incluyendo el método main y uso de funciones auxiliares.

Ejercicio 19: Convertir un vector a una cadena de texto con el formato (x, y).

En este programa implementé una función para convertir un vector representado como un arreglo en una cadena de texto legible, con el formato matemático habitual (x, y). Para ello:

Definí un método estático llamado vectorToString, que recibe como parámetro un arreglo double[] de dos posiciones, representando un vector bidimensional.

Dentro del método, concatené las dos componentes del vector con paréntesis y una coma para formar una cadena con el formato (x, y).

En el método main, declaré un vector v con los valores 3.5, 4.2.

Llamé al método vectorToString(v) y usé System.out.println para mostrar la cadena generada por la función.

Arreglos de tipo double[] para representar el vector 2D.

Concatenación de cadenas usando el operador + para formar la representación del vector.

Un método estático auxiliar (vectorToString) para encapsular la lógica de conversión.

Salida por consola con System.out.println para mostrar el vector convertido en texto.

Ejercicio 20: Calcular el promedio de un conjunto de vectores.

En este programa implementé una función para calcular el promedio de varios vectores bidimensionales, representados como arreglos.

Definí una función llamada promedio, que recibe un arreglo bidimensional double[][], donde cada subarreglo representa un vector 2D.

Declaré dos variables sumX y sumY para acumular las componentes x y y de todos los vectores.

Utilicé un bucle for-each para recorrer cada vector del arreglo y sumar sus componentes respectivas.

Al final, dividí cada suma entre el número total de vectores (vectores.length) para obtener el promedio de las componentes x e y.

La función devuelve un nuevo vector double[] que representa el vector promedio.

En el método main, declaré tres vectores bidimensionales: 2, 4, 4, 6 y 6, 8, los pasé a la función promedio y luego imprimí el resultado con System.out.println. Arreglo bidimensional double[][] para representar una colección de vectores 2D.

Bucle for-each para recorrer cada vector del arreglo.

Acumuladores (sumX, sumY) para sumar los valores de cada componente.

Operación de promedio (suma / cantidad) para calcular el vector medio.

Creación de un nuevo vector (double[]) para devolver el resultado.

Impresión por consola usando System.out.println.

2.8 Habilidades blandas empleadas en la práctica

- ☐ Liderazgo
- ☐ Trabajo en equipo
- ☐ Comunicación asertiva
- ☐ La empatía
- ☐ Pensamiento crítico
- ☐ Flexibilidad
- ☐ La resolución de conflictos
- ☐ Adaptabilidad
- ☐ Responsabilidad

2.9 Conclusiones

La utilización de VECTORES permite reutilizar código en los programas.

2.10 Recomendaciones

El uso de arrays en Java es fundamental para la manipulación eficiente de conjuntos de datos. A través de los ejercicios trabajados, se ha demostrado que los arrays permiten almacenar, acceder y procesar múltiples valores de forma ordenada y estructurada. Su uso facilita la implementación de algoritmos que requieren repetición, búsqueda, ordenamiento o análisis de datos.

2.11 Referencias

Insertar las referencias bibliográficas empleadas aplicando la norma IEEE.

Deitel, P. J., Deitel, H. M. (2017). Java: How to Program (Early Objects) (11th Edition). Pearson Education.

2.12 Anexos

https://github.com/mrovalino/Ejercicios-con-vectores.git