Práctica 4

Gabriel Alberto Barrios de León - 201804558[1]

[1]*Escuela de Ciencias Físicas y Matemáticas, Universidad de San Carlos, Ciudad Universitaria, Zona 12, Guatemala.*

I. OBJETIVOS:

A. Objetivos Generales:

1. Analizar, crear soluciones y diagramas de flujo para programas en lenguaje C.

B. Objetivos Específicos:

1. Implementar una solución numérica ante el problema que se le presente.
2. Crear la documentacion necesaria para cada programa elaborado.
3. Aplicar los conocimientos adquiridos sobre el lenguaje C.

II. MARCO TEÓRICO:

A. Problema 1:

1. Enunciado del problema:

Debe de ingresar un vector de 10 elementos, llenarlo de numeros pares del 2 al 20. Al iniciar el programa debe preguntar al usuario como quiere ver los numeros, el menú debe de ser por medio de caracteres: "a"verlos de forma ascendente, "d" descendente, en caso que el usuario escriba otro valor debe de decir que no es correcto y preguntarle el caracter nuevamente, hasta que este sea el correcto, al ingresar el valor correcto muestra el vector en pantalla y termina el programa.

2. Metodología:

Hay una sola forma posible de ordenar el vector, así que lo definimos dentro del código. Se pregunta al usuario cómo quiere ver el vector, y se guarda la respuesta en una variable. Mediante la confición if se determina cómo imprimir el vector, si se imprime mediante espacios o saltos en línea.

3. Entradas y Salidas:

Entradas: a o d
Salias: vector vertical u horizontal. Mensaje para el usuario.

4. Diagrama de flujo:

Ver anexos.

B. Problema 2:

1. Enunciado del problema:

Crear un programa que solicite al usuario 5 números enteros. al terminar de guardar los valores, el programa debe de ordenarlos de forma ascendente y mostrar el vector ordenado.

2. Metodología:

Se imprime un mensaje solicitando 5 numeros enteros, y estos se guardan en un vector mediante scanf. Recurrimos al método de ordenamiento burbuja, en el cual usamos ciclos while para recorrer el vector 25 veces $(5 * 5)$. Se empieza por el primer par de casillas, de izquierda a derecha. Si el número de la izquierda es mayor, este se guarda y en su lugar se coloca el número de la derecha, para después colocar en la segunda casilla el valor que fue guardado. Luego, se procede con el segundo par (casillas 1 y 2) y se repite el proceso. En caso de que el número a la derecha sea mayor al de la izquierda, este par se ignora y se procede con el siguiente; al terminar el vector, se repite el proceso hasta obtener el vector ordenado de forma ascendente. Este último se imprime.

3. Entradas y Salidas:

Entradas: 5 números enteros.
Salidas: números ordenados de forma ascendente.

4. Pseudo Código:

1. Inicio
2. Declaramos n=4;x,i, interruptor, pasada y j, todos igualados a 0.
3. Declaramos un vector v[n] con 5 casillas.
4. Pedimos al usuario 5 números enteros.
5. Leemos los valores.
6. Guardamos los números en v[].

7. Mientras $pasada = 0; pasada < ninterruptor$, se fija interruptor igual a 0. Mientras $j < n - pasada$:
8. —-Si $A[j] > A[j+1]$, entonces declaramos auxiliar= $A[j]$, $A[j] = A[j+1]$, $A[j+1] = aux$.
9. —-Interuptor igual a 1.
10. Imprime vector A[] ordenado.
11. Fin.

C. Problema 3:

1. Enunciado del problema:

Crear un programa que solicite al usuario dos posiciones en coordenadas (x,y,z) al obtenerlas debe de almacenarlas en dos vectores, el programa automaticamente debe de mostrar los siguientes resultados: magnitud, suma, producto escalar y producto vectorial.

2. Metodología:

Mediante un mensaje se solicitan 6 números que se almacenarán en dos vectores a y b mediante scanf. Sea v un vector. Su magnitud es dada por $|v|^2 = x^2 + y^2 + z^2$, de modo que se realiza una función que multiplica las componentes de dos vectores y las suma: ingresamos dos veces el mismo vector para obtener su magnitud, e ingresamos vectores distintos para el producto escalar.

Una función de suma obtiene las componentes de ambos vectores y las suma ordenadamente, regresando un vector. Para el producto vectorial, se sigue la siguiente fórmula:

$$\vec{a}x\vec{b} = (y_1z_2 - z_1y_2)\vec{x} - (x_1z_2 - x_2z_1)\vec{y} + (x_1y_2 - x_2y_1)\vec{z}$$

3. Entradas y Salidas:

Entradas: 6 números
Salidas: magnitud, suma, producto escalar y producto vectorial.

4. Diagrama de flujo:

Ver anexos.

D. Problema 4:

1. Enunciado del problema:

Crear un programa que solicite al usuario dos matrices de 3X3 almacenarlas como (matA, matB) y una constante, el programa automaticamente debe de mostrar los siguientes resultados, aprovechando los arreglos matriciales con dos ciclos for:

1. matA por constante.
2. Suma de las dos matrices.
3. Multiplicación de las dos matrices.
4. Determinante de matA.
5. Traspuesta de matB.
6. Inversa de matA.
7. Reducción de Gauss.
8. Reducción Gauss-Jordan.

2. Metodología:

Se solicitan 18 números, y el código se encarga de lo siguiente según el listado anterior.

1. Se multiplican todas las componentes por la constante.
2. Se suman las componentes de A con las componentes respectivas de B.
3. Se usa el criterio fila por columna para determinar cada componente de una matriz llamada "producto".
4.

$$detmatA = a11*(a22*a33-a23*a32)-a12*(a21*a33-a23*a31)$$

$$+a13*(a21*a32-a22*a31)$$

5. Para la traspuesta cambiamos los índices de la matriz matB ij por ji, dejando la diagonal invariante.
6. Para la inversa calculamos a mano la matriz adjunta de matA y su traspuesta. Este resultado lo dividimos entre el determinante:

$$matA^{-1} = \frac{(adj(matA)^T}{det(matA)}$$

7. Para Gauss igualamos virtualmente la matriz a un vector columna lleno de ceros. Tendremos matrices auxiliares donde guardar información. Dadas las componentes a21 y a11, multiplicamos fila uno por a21 y fila dos por a11 y guardamos esta información. restamos filas y obtenemos un cero en a21. Repetimos el proceso con a11 y a31, y obtenemos otro cero en a31. Repetimos esto para a22 y a33 con el fin de obtener un cero en a32. Finalmente, se imprime la matriz resultante (que es una triangular) y un vector vertical con nuevas componentes.

8. Se ejecuta el proceso de reducción de Gauss ya mencionado, y se continúa para la matriz triangular superior, de modo que el proceso se repite para a13 y a23, y se obtiene un cero en a13; se repite para a12 y a22 y se obtiene otro cero en a12; se repite para a33 y a23, y se obtiene un sexto cero en a23. Finalmente, se dividen las filas por el único número que ha quedado para obtener una matriz diagonal llena de unos igualada a un vector columna. Se imprimen las componentes de este vector.

3. Entradas y Salidas:

Se ingresan 9 números para matA y 9 para matB, además de una constante.
Salidas: 8 operaciones con matrices.

4. Pseudocódigo:

1. Inicio
2. Flotamos dos matrices de 3x3,matA y matB.
3. Flotamos una matriz de 3x3 llena de ceros llamada "negativo".
4. Declaramos fila y columna.
5. Pedimos al usuario introducir 9 números, que almacenamos en matA.
6. Pedimos al usuario introducir 9 números, que almacenamos en matB.
7. Pedimos al usuario una constante y la guardamos.
8. Flotamos otra matriz de 3x3 llena de ceros matc.
9. Para todo elemento de matA le multiplicamos la constante y lo guardamos en matC.
10. Imprimir matC.
11. Suma(matA,matB).
12. Para todo elementeo de matB lo multiplicamos por -1 y lo guardamos en "negativo"
13. Suma(matA,negativo).
14. Flotamos una matriz "producto".
15. Por cada elemento de matA
16. —-Declaramos sum.
17. —-Sea j las columnas.
18. —-Mientras j sea menor a la dimensión de las matrices:
19. ———$sum+ = matA[i][j] * matB[j][a]$.
20. —$producto[i][a] = sum$
21. Imprimir producto.
22. flotaomos primero, segundo, tercero, det.
23. $primero = matA[0][0] * (matA[1][1] * matA[2][2] - matA[1][2] * matA[2][1])$.
24. $segundo = matA[0][1] * (matA[1][2] * matA[2][0] - matA[1][0] * matA[2][2])$.
25. $tercero = matA[0][2] * (matA[1][0] * matA[2][1] - matA[1][1] * matA[2][0])$.
26. $det = primero + segundo + tercero$.
27. Imprimir det.
28. Flotamos traspuesta, cofactores, inversa.
29. $traspuesta[fila][columna] = matB[columna][fila]$.
30. Imprimir traspuesta.
31. Calcular matriz de cofactores.
32. Dividir cada elmento de cofactores entre detA y guardarlo en inversa.
33. Imprimir inversa.
34. Fin

1. Inicio impresion.
2. Declaramos fila, columna.
3. Mientras $fila < 3$.
4. —-Mientras $columna < 3$
5. ———Imprimir $matriz[fila][columna]$
6. Fin.

1. Inicio Suma
2. Declaramos sumita.
3. Mientras $fila < 3$.
4. —-Mientras $columna < 3$
5. ———sumita$[fila][col] = matriz[fila][col] + matroz[fila][col]$.
6. Fin.

E. Problema 5:

1. Enunciado del problema:

Crear un programa que encuentre el factorial de un numero entero ingresado, debe de utilizar una funcion recursiva.

2. Metodología:

Se define: $n! = n(n-1)...2*1$. Se usa una función que se llama así misma para multiplicar números enteros partiendo del 1 hasta llegar a n.

3. Entradas y Salidas:

Entrada: número entero n. Salida: $n!$.

4. Diagrama de flujo:

Ver anexos.

F. Problema 6:

1. Enunciado del problema:

Crear un programa que realice la sumatoria desde 1 hasta un numero n que ingrese el usuario de las siguientes funciones:

$$\sum_{i=1}^{n} k^2(k-3)$$

$$\sum_{i=1}^{n} \frac{3}{k-1}$$

$$\sum_{i=1}^{n} \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

$$\sum_{i=1}^{n} 0.1(3*2^{k-2}+4)$$

2. Metodología:

Mediante scanf se obtiene un numero entero. Se hacen cuatro funciones de suma con una variable flotante a la que se le va sumando el valor de la suma deseada con diferentes enteros en un ciclo hasta llegar al número elegido. Se imprime el resultadode las cuatro sumatorias.

3. Entradas y Salidas:

Entrada: número entero.
Salida: el resultadod e cuatro sumatorias desde 1 al número elegido. Nótese que el código no colapsa para $k = 1$, pues parece anularse a falta de memoria. Entonces es lo mismo empezar con $i = 2$.

4. Pseudocódigo:

1. Inicio
2. Declaramos variables: n.
3. Pedimos al usuario un número entero.
4. Guardamos el numero en n.
5. Evaluamos n en suma1().
6. Evaluamos n en suma2().
7. Evaluamos n en suma3().
8. Evaluamos n en suma4().
9. Fin

1. Inicio suma1().
2. Declaramos i, k, ans igualados a cero.
3. Mientras $i <= k$:
4. —$ans+ = pow(i, 2) * (i - 3)$.
5. Imprimir ans.
6. Fin

1. Inicio suma1().
2. Declaramos i= 1, k, ans igualados a cero.
3. Mientras $i <= k$:
4. —$ans+ = 3/(k - 1)$.
5. Imprimir ans.
6. Fin

1. Inicio suma1().
2. Declaramos k, i= 1.
3. Declaramos momo y momo 2.
4. Declaramos como doubles: $result, numero = 5, mitad = 0.5$.
5. $result = pow(numero, mitad)$
6. $momo = (1 + result)/2$.
7. $momo2 = (1 - result)/2$.
8. Mientras $i <= k$:
9. —$ans+ = result * (pow(momo, i) - pow(momo2, i))$.

10. Imprimir ans.
11. Fin

1. Inicio suma1().
2. Declaramos i= 1, k, ans igualados a cero.
3. Declaramos uno= 0.1
4. Mientras $i <= k$:
5. —$ans+ = uno * (4 + 3 * pow(2, i))$.
6. Imprimir ans.
7. Fin

III. ANEXOS

Diagrama de flujo para el primer ejercicio:

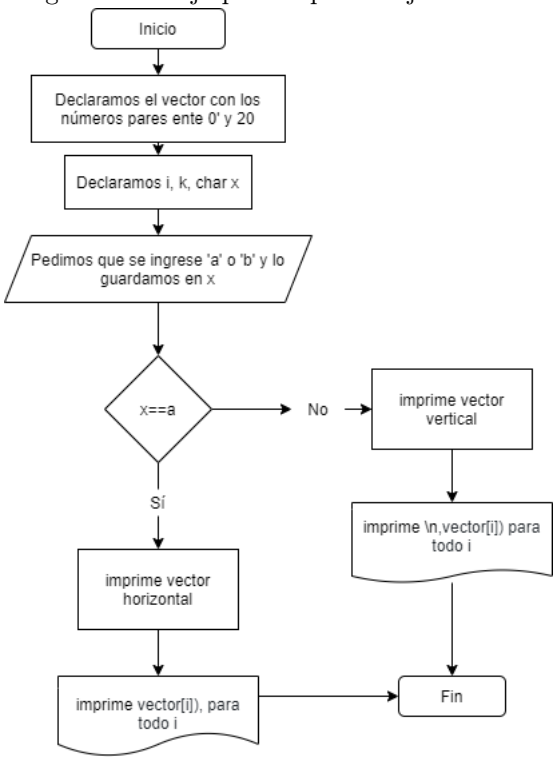

Diagrama de flujo para el tercer ejercicio:

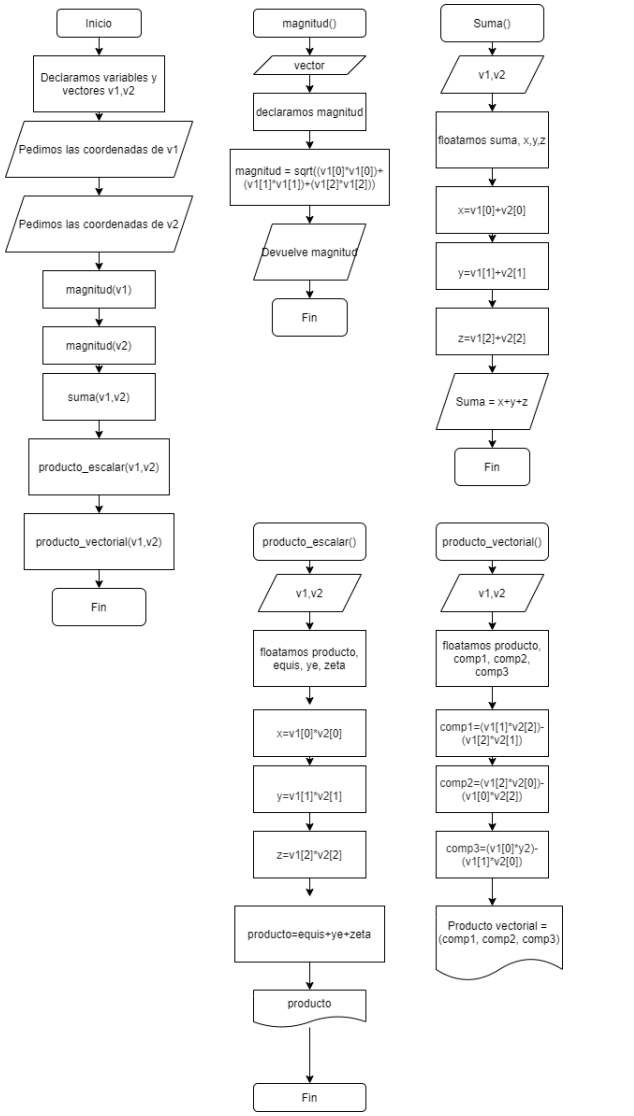

Diagrama de flujo para el quinto ejercicio: