INGENIERIA EN COMPUTACION

Materia: Programación Estructurada

Alumno: Ayon Dimas Andres

Matricula: 366845

Profesor: Pedro Nunes Yepiz

Actividad: #8

Tema: Vectores y Matrices

Ensenada B.C 3 de octubre del 2023

INTRODUCCION

En esta práctica ya podremos ver lo que son los arreglos y sobretodo las matrices, que hasta a el momento solo serán matrices de 4x4, así mismo podremos ver cómo es que haremos funciones para poder realizar tanto los vectores como las matrices, utilizando las sentencias de prácticas pasadas como lo son los ciclos y las condiciones, que es lo que venimos trabajando alrededor de estas 8 practicas donde hemos aprendido a implementar condiciones en ciclos, funciones, arreglos, etc. Podremos seguir viendo esto y para qué sirven los vectores y las matrices.

COMPETENCIA

El alumno deberá aprender a dominar estas nuevas sentencias que son los vectores y matrices de 4x4, sin dejar de lado lo aprendido en anteriores prácticas, ya que lo utilizado anteriormente nos seguirá siendo útil para poder realizar esta práctica y es de suma importancia saber implementar todas y cada una de estas sentencias en C para su posterior uso en otros programas más avanzados en los que utilizaremos nuevas sentencias.

FUNDAMENTOS

Funciones

Para entender esto podemos decir que El código de un programa escrito en C se divide en funciones. Aunque similares a los “métodos” de Java, las funciones no están asignadas ni a una clase ni a un objeto. Una función en C se distingue *sólo* por su nombre. Dos funciones con igual nombre y con diferente número y tipo de parámetros se considera una definición múltiple, y por tanto un error.

Las funciones suelen encapsular una operación más o menos compleja de la que se deriva un resultado. Para ejecutar esta operación, las funciones pueden precisar la invocación de otras funciones (o incluso de ellas mismas como es el caso de las funciones recursivas).

Las funciones en un programa son entidades que dado un conjunto de datos (los parámetros), se les encarga realizar una tarea muy concreta y se espera hasta obtener el resultado. Lo idóneo es dividir tareas complejas en porciones más simples que se implementan como funciones. La división y agrupación de tareas en funciones es uno de los aspectos más importantes en el diseño de un programa.

Arreglos y matrices

Para empezar, podemos decir que en C a los vectores también se les llama arrays o arreglos

- Las matrices serán vectores de vectores

Los arrays son:

– Conjuntos de variables del mismo tipo

– El cual tienen el mismo nombre.

– Se diferencian en el índice

Es una manera de manejar una gran cantidad de datos del mismo tipo bajo un mismo nombre o Identificador.

Para realizar operaciones matemáticas sobre un array (como en Matlab) debemos operar sobre cada elemento del array.

Arrays de una dimensión

Un array (unidimensional, también denominado vector) es una variable estructurada formada de un número "n" de variables simples del mismo tipo que son denominadas los componentes o elementos del array. El número de componentes "n" es, entonces, la dimensión del array. De igual manera que en matemáticas, decimos que "A" es un vector de dimensión "n".

Arrays de dos dimensiones

Un array en C puede tener una, dos o más dimensiones. Por ejemplo, un array de dos dimensiones también denominado matriz, es interpretado como un array (unidimensional) de dimensión "f" (número de filas), donde cada componente es un array (unidimensional) de dimensión "c" (número de columnas). Un array de dos dimensiones, contiene, pues, "f*c" componentes.

PROCEDIMIENTO

Arreglo unidimensional

```
#include <stdio.h>

// Declaración de funciones
void llenarVector(int vector[], int tamano);
void imprimirVector(int vector[], int tamano);
int sumarVector(int vector[], int tamano);

int main() {
    int tamano;
    printf("Ingrese el tamaño del vector: ");
    scanf("%d", &tamano);

    // Crear un vector de tamaño 'tamano'
    int miVector[tamano];

    // Llenar el vector con valores ingresados por el usuario
    llenarVector(miVector, tamano);

    // Imprimir el vector
    printf("El vector ingresado es:\n");
    imprimirVector(miVector, tamano);

    // Calcular y mostrar la suma de los elementos del vector
```

```
    int suma = sumarVector(miVector, tamano);
    printf("La suma de los elementos del vector es: %d\n", suma);

    return 0;
}

// Definición de la función para llenar el vector
void llenarVector(int vector[], int tamano) {
    printf("Ingrese los elementos del vector:\n");
    for (int i = 0; i < tamano; i++) {
        scanf("%d", &vector[i]);
    }
}

// Definición de la función para imprimir el vector
void imprimirVector(int vector[], int tamano) {
    for (int i = 0; i < tamano; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");
}

// Definición de la función para sumar los elementos del vector
int sumarVector(int vector[], int tamano) {
    int suma = 0;
    for (int i = 0; i < tamano; i++) {
        suma += vector[i];
```

```
    }
    return suma;
}
```

Ejemplo de matriz en C

```
#include <stdio.h>

// Constante para el tamaño de la matriz
#define FILAS 4
#define COLUMNAS 4

// Declaración de funciones
void llenarMatriz(int matriz[FILAS][COLUMNAS]);
void imprimirMatriz(int matriz[FILAS][COLUMNAS]);
int sumarMatriz(int matriz[FILAS][COLUMNAS]);

int main() {
    int miMatriz[FILAS][COLUMNAS];

    // Llenar la matriz con valores ingresados por el usuario
    llenarMatriz(miMatriz);

    // Imprimir la matriz
    printf("La matriz ingresada es:\n");
    imprimirMatriz(miMatriz);
```

```
    // Calcular y mostrar la suma de los elementos de la matriz
    int suma = sumarMatriz(miMatriz);
    printf("La suma de los elementos de la matriz es: %d\n", suma);

    return 0;
}

// Definición de la función para llenar la matriz
void llenarMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("Ingrese los elementos de la matriz (%dx%d):\n", FILAS, COLUMNAS);
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            scanf("%d", &matriz[i][j]);
        }
    }
}

// Definición de la función para imprimir la matriz
void imprimirMatriz(int matriz[FILAS][COLUMNAS]) {
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }
}
```

```
// Definición de la función para sumar los elementos de la matriz
int sumarMatriz(int matriz[FILAS][COLUMNAS]) {
    int suma = 0;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            suma += matriz[i][j];
        }
    }
    return suma;
}
```

RESULTADOS Y CONCLUSIONES

El uso de funciones, arreglos y matrices en programación es algo esencial a la hora de hacer un programa que sea eficiente y fácil de comprender. Las funciones permiten dividir el código en unidades lógicas y reutilizables, lo que facilita la comprensión y la solución de problemas complejos. Los arreglos y matrices son estructuras de datos que nos permiten almacenar y manipular conjuntos de elementos de manera organizada.

Las funciones nos permiten utilizar solo cierta parte del código para realizar tareas específicas en bloques de código separados, lo que promueve la reutilización y la legibilidad. Además, facilitan la colaboración en equipos de desarrollo al dividir el trabajo en módulos independientes. Los arreglos unidimensionales nos permiten almacenar una colección ordenada de elementos del mismo tipo, como una lista de números. Los arreglos bidimensionales, o matrices, agregan una dimensión adicional y se utilizan para representar estructuras tabulares de datos, como matrices numéricas o tablas. Al utilizar estas herramientas en conjunto, podemos crear programas más poderosos y versátiles. Por ejemplo, en el caso de matrices, podemos realizar operaciones matriciales y resolver problemas complejos de manera eficiente.

En resumen, el uso de funciones, arreglos y matrices es fundamental en la programación para organizar el código, mejorar la legibilidad y resolver una amplia variedad de

problemas. Dominar estas herramientas es esencial para convertirse en un programador eficiente y efectivo.

REFERENCIAS

https://www.it.uc3m.es/pbasanta/asng/course_notes/functions_es.html

https://www.uco.es/grupos/eatco/informatica/metodologia/cadenasyarrays.pdf