



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

Profesor: M.I. Oscar René Valdez Casillas

Fundamentos de Programación

Práctica de estudio 09: Arreglos unidimensionales.

Integrantes:

Arteaga Gonzalez Jaime Alejandro

Márquez Alan

Valencia Francisco Valentín

García Jiménez Joel David

Grupo: 31

Arteaga, J., Márquez, A., García, J. y Valencia, V., (2024). Práctica de estudio 09: Arreglos unidimensionales.

Índice:

1. Resumen.....	
.....	2
2. Introducción.....	
.....	2
3. Objetivo.....	
.....	2
4. Ejercicios a realizar.....	
...2,3,4,5,6	
• Ejercicio 1. Programa que reciba cuatro puntos.	
• Ejercicio 2. Programa que permita realizar el producto punto de dos vectores definido.	
• Ejercicio 3. Programa que almacena las 10 calificaciones de un estudiante en un arreglo y obtenga el promedio de todas las calificaciones que son distintas de cero.	
5. Conclusiones.....	
.....	6,7
• Reflexión individual de los integrantes del equipo.	
6. Bibliografía.....	
.....	7
• Fuentes y referencias utilizadas en el informe.	

Resumen.

En el presente reporte, elaboramos programas que contienen arreglos, de manera que el entendimiento de estos sea más claro y conciso, para tener un mejor desempeño a la hora de programar, los arreglos unidimensionales en estos programas fueron útiles ya que , son una estructura de datos que permite almacenar y manipular un conjunto de elementos del mismo tipo, como lo haremos en cada uno de los ejercicios mostrados en los que requieran agrupar datos del mismo tipo, alineados en un vector o lista.

Introducción.

El uso de arreglos unidimensionales es fundamental en programación, ya que permite el uso de datos del mismo tipo de manera ordenada y eficiente. Este objetivo buscará que aprendan a implementar y manipular arreglos para resolver problemas prácticos, como gestionar listas de estudiantes o puntuaciones. A través de la elaboración de programas, los estudiantes desarrollarán habilidades en el manejo de colecciones de datos, fomentando un entendimiento más profundo de la programación y su aplicabilidad en diversas situaciones. Al finalizar, estarán mejor preparados para enfrentar desafíos y diseñar soluciones efectivas.

Objetivo.

El alumno utilizará arreglos de una dimensión en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, alineados en un vector o lista.

Ejercicios a realizar.

Ejercicio 1. Realizar un programa que reciba cuatro puntos en R3 y calcule la distancia que hay del primero a los otros tres puntos.

Lectura 1: El usuario ingresa las coordenadas x, y, z, para cada punto.

Lectura 2: Se utiliza una función cd para calcular la distancia que toma dos puntos y calcula la distancia euclidiana entre ellos.

Lectura 3: Finalmente, el programa calcula y muestra la distancia desde el primer punto a los otros tres.

```
#include <stdio.h>
```

```

#include <math.h>

double cd(double p1[3], double p2[3]) {
    return sqrt(pow(p2[0] - p1[0], 2) + pow(p2[1] - p1[1], 2) + pow(p2[2] - p1[2], 2));
}

int main() {
    double p[4][3];
    int i, j;
    for (i = 0; i < 4; i++) {
        printf("Introduce las coordenadas del punto %d (x y z): ", i+1);
        for (j = 0; j < 3; j++) {
            scanf("%lf", &p[i][j]);
        }
    }
    for (i = 1; i < 4; i++) {
        double d = cd(puntos[0], puntos[i]);
        printf("Distancia desde el punto 1 al punto %d: %.2f\n", i+1, d);
    }
    return 0;
}

```

Pseudocódigo:

INICIO

Definir función cd(punto1, punto2):

Recibir como parámetros dos arreglos de 3 elementos (coordenadas de los puntos en).

Calcular la distancia euclidiana entre punto1 y punto2.

Retornar la distancia calculada.

Definir un arreglo p[4][3] para almacenar las coordenadas de los 4 puntos.

Para i desde 0 hasta 3 hacer:

Mostrar el mensaje: "Introduce las coordenadas del punto i+1 (x y z)".

Para j desde 0 hasta 2 hacer:

Leer p[i][j] (la coordenada correspondiente de cada punto).

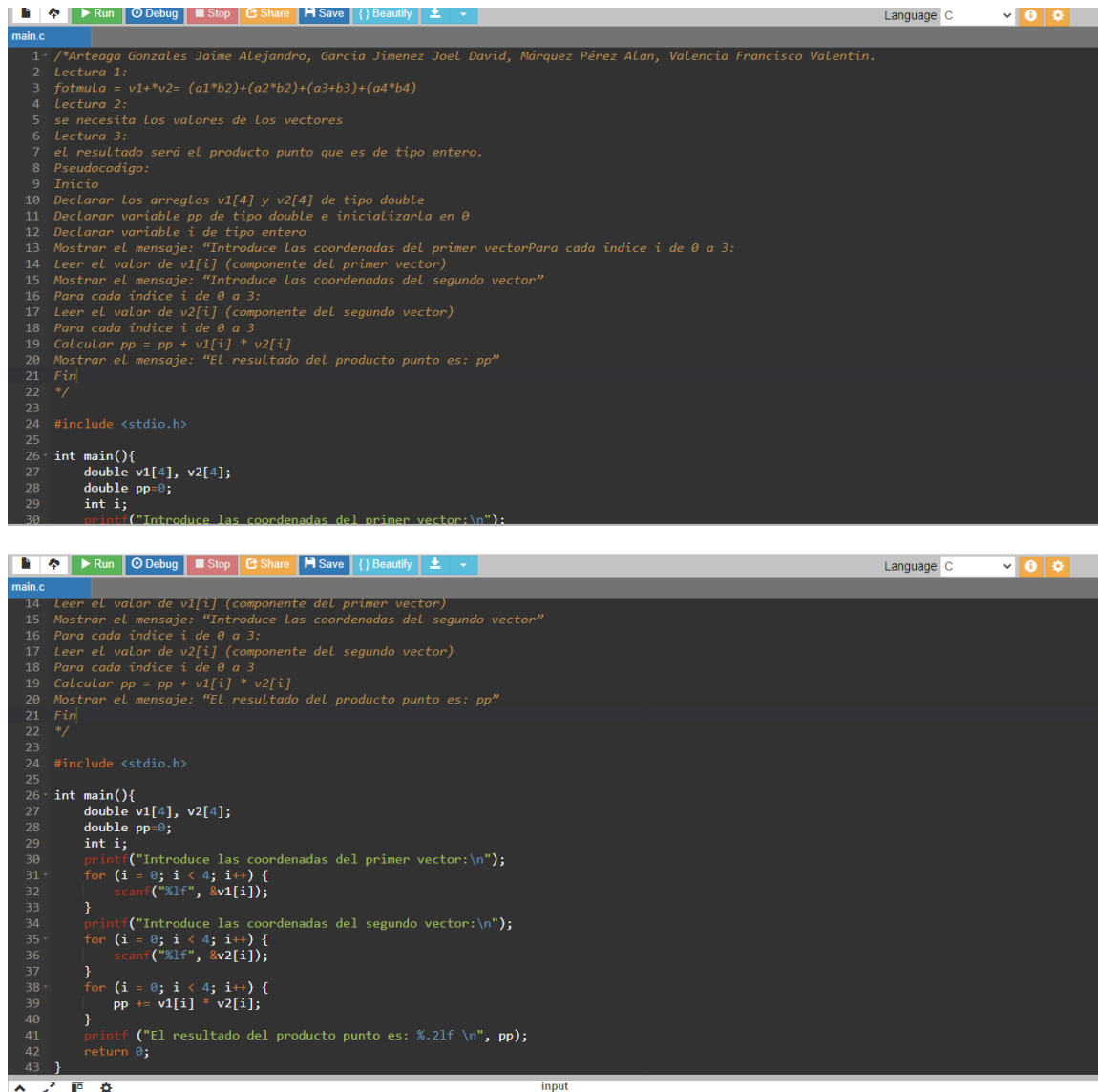
Para i desde 1 hasta 3 hacer:

Llamar a la función cd para calcular la distancia entre p[0] y p[i].

Mostrar el resultado: "Distancia desde el punto 1 al punto i+1".

FIN

Ejercicio 2. Realizar un programa que permita realizar el producto punto de dos vectores definidos en R4.



```
1- /*Arteaga Gonzales Jaime Alejandro, Garcia Jimenez Joel David, Márquez Pérez Alan, Valencia Francisco Valentin.
2- Lectura 1:
3- formula = v1*v2= (a1*b2)+(a2*b2)+(a3*b3)+(a4*b4)
4- Lectura 2:
5- se necesita los valores de los vectores
6- Lectura 3:
7- el resultado será el producto punto que es de tipo entero.
8- Pseudocodigo:
9- Inicio
10- Declarar los arreglos v1[4] y v2[4] de tipo double
11- Declarar variable pp de tipo double e inicializarla en 0
12- Declarar variable i de tipo entero
13- Mostrar el mensaje: "Introduce las coordenadas del primer vectorPara cada índice i de 0 a 3:
14- Leer el valor de v1[i] (componente del primer vector)
15- Mostrar el mensaje: "Introduce las coordenadas del segundo vector"
16- Para cada índice i de 0 a 3:
17- Leer el valor de v2[i] (componente del segundo vector)
18- Para cada índice i de 0 a 3
19- Calcular pp = pp + v1[i] * v2[i]
20- Mostrar el mensaje: "El resultado del producto punto es: pp"
21- Fin
22- */
23-
24- #include <stdio.h>
25-
26- int main(){
27-     double v1[4], v2[4];
28-     double pp=0;
29-     int i;
30-     printf("Introduce las coordenadas del primer vector:\n");
14- Leer el valor de v1[i] (componente del primer vector)
15- Mostrar el mensaje: "Introduce las coordenadas del segundo vector"
16- Para cada índice i de 0 a 3:
17- Leer el valor de v2[i] (componente del segundo vector)
18- Para cada índice i de 0 a 3
19- Calcular pp = pp + v1[i] * v2[i]
20- Mostrar el mensaje: "El resultado del producto punto es: pp"
21- Fin
22- */
23-
24- #include <stdio.h>
25-
26- int main(){
27-     double v1[4], v2[4];
28-     double pp=0;
29-     int i;
30-     printf("Introduce las coordenadas del primer vector:\n");
31-     for (i = 0; i < 4; i++) {
32-         scanf("%lf", &v1[i]);
33-     }
34-     printf("Introduce las coordenadas del segundo vector:\n");
35-     for (i = 0; i < 4; i++) {
36-         scanf("%lf", &v2[i]);
37-     }
38-     for (i = 0; i < 4; i++) {
39-         pp += v1[i] * v2[i];
40-     }
41-     printf("El resultado del producto punto es: %.2lf \n", pp);
42-     return 0;
43- }
```

<https://onlinegdb.com/EdZMF47dJ>

```

Introduce las coordenadas del primer vector:
1
2
3
4
Introduce las coordenadas del segundo vector:
5
6
7
8
El resultado del producto punto es: 70.00

...Program finished with exit code 0
Press ENTER to exit console

```

Ejercicio 3: Realizar un programa que almacene las 10 calificaciones de un estudiante en un arreglo y obtenga el promedio de todas las calificaciones que son distintas de cero. Se deberá seguir realizando el cálculo hasta que se indique que ya no se quiere evaluar a ningún estudiante

```

1 ▾ /*Lectura 1: AlmaceneNAR las 10 calificaciones de un estudiante
2 Arreglo y obtenga el promedio de todas las calificaciones
3 Distintas de cero
4 ▾ /*Lectura 2: Entradas
5 i y op: Enteros
6 promedio, suma, calificacion: Flotantes
7 ▾ /*Lectura 3: Salidas
8 El promedio del alumno es: promedio
9 Quieres continuar con otro alumno: Si, No.
10 Promedio del alumno: Flotante

```

```

1- /*INICIO
2
3  DECLARSR i, op COMO ENTERO
4  DECLARAR prom, suma COMO REAL
5  DECLARAR cali[10] COMO REAL
6
7  op = 0
8
9  MIENTRAS (op != 0) HACER
10     suma = 0
11     prom = 0
12
13     PARA i DESDE 0 HASTA 9 HACER
14         IMPRIMIR "Escribe la calificacion [", i + 1, "]: "
15         LEER cali[i]
16
17         MIENTRAS (cali[i] <= 0) HACER
18             IMPRIMIR "Valor no valido, ingrese nuevamente: "
19             LEER cali[i]
20         FIN MIENTRAS
21
22     suma = suma + cali[i]
23 FIN PARA
24
25 prom = suma / 10
26 IMPRIMIR "El promedio del alumno es: ", prom
27
28 IMPRIMIR "¿Quieres continuar con otro alumno? (Si 1, No 0): "
29 LEER op
30 IMPRIMIR "\n"
31 FIN MIENTRAS
32
33 RETORNAR 0
34 FIN */
35

```

```

36 #include <stdio.h>
37 int main(){
38 int i=0, op=0;
39 float prom=0, suma=0, cali[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
40 do {
41 suma = 0;
42 prom = 0;
43 for(i=0; i<10; i++){
44 printf("Escribe la calificacion [%d]: ", i+1 );
45 scanf("%f", &cali[i]);
46 while (cali [i] <= 0){
47 printf("Valor no valido, ingrese nuevamente: \n");
48 scanf("%f", &cali[i]);
49 }
50 suma += cali[i];
51 prom = suma/10;
52 }
53 printf("\n");
54 printf("El promedio del alumno es: %f", prom);
55 printf("\n¿Quieres continuar con otro alumno?\nSi 1\nNo 0\n");
56 scanf("%d", &op);
57 printf("\n");
58 } while(op != 0);
59 return 0;
60 }

```

```
Escribe la calificacion [1]: 10
Escribe la calificacion [2]: 9
Escribe la calificacion [3]: 8
Escribe la calificacion [4]: 9
Escribe la calificacion [5]: 7
Escribe la calificacion [6]: 8
Escribe la calificacion [7]: 8
Escribe la calificacion [8]: 7
Escribe la calificacion [9]: 9
Escribe la calificacion [10]: 9

El promedio del alumno es: 8.400000
¿Quieres continuar con otro alumno?
Si 1
No 0

```

Conclusiones:

Joel: Utilizamos arreglos de una dimensión para resolver problemas que requirieron agrupar datos del mismo tipo, alineados en un vector o lista. Esta habilidad no solo facilitará la organización y manipulación de datos, sino que también permite abordar problemas de manera más eficiente y estructurada. De ahora en adelante considero que estaremos mejor preparados para desarrollar soluciones prácticas y efectivas.

Jaime: Los arreglos para mi en lo personal era de difícil comprensión ya que es un concepto nuevo para mi y aunque ya anteriormente había programado, los arreglos son cosas nuevas que me han dado en los fundamentos de programación,, esta práctica tuvo ese objetivo, ya que pusimos en práctica estos arreglos en cada uno de los ejercicios mostrados. Los arreglos en cada uno de los ejercicios nos permiten almacenar y acceder a un conjunto de elementos del mismo tipo en una ubicación de memoria contigua, de esta manera puedo concluir esta práctica diciendo que hemos avanzado mucho a lo largo de todas las prácticas y llegamos a los arreglos que aunque fue difícil su comprensión, poco a poco nos va a facilitar las acciones y los almacenados a la hora de programar.

Alan: Pusimos en práctica los arreglos para hacer un ejercicio de vectores que en lo personal se me dificultó un poco. También el uso de arreglos almacenan los valores de las coordenadas de los vectores, mientras que el ciclo for facilita la iteración sobre sus componentes para realizar las multiplicaciones correspondientes. Al final, acumulamos el resultado de estas multiplicaciones en una variable para obtener el producto punto.

Este tipo de problemas resalta la importancia de cómo estructurar un programa sencillo que involucra entrada de datos, cálculos matemáticos básicos y la salida de un resultado.

Valentín:

BIBLIOGRAFÍAS:

De Granda, D. (2022, 3 noviembre). *Arrays o arreglos*.

/Clases/3208-programacion-basica/51964-arrays-o-arreglos/.

<https://platzi.com/clases/3208-programacion-basica/51964-arrays-o-arreglos/#:~:text=En%20programaci%C3%B3n%2C%20un%20array%20o,una%20ubicaci%C3%B3n%20de%20memoria%20contigua.&text=Cada%20elemento%20del%20array%20tiene,a%20%C3%A9l%20utilizando%20su%20%C3%ADndice.>

Aprendiendo Arreglos (Arrays) Unidimensionales en PSeInt | Blog Coders Free. (s. f.-b).

Coders Free.

<https://codersfree.com/posts/aprendiendo-arreglos-unidimensionales-en-pseint>

UNIDIMENSIONAL. (s. f.).

<https://www.uacj.mx/CGTI/CDTE/JPM/Documents/IIT/arreglos1/unidimensional.html#:~:text=Son%20los%20que%20permiten%20acceder,la%20posici%C3%B3n%20dentro%20del%20arreglo.>