ALGORITMOS Y ESTRUCTURAS DE DATOS - 2023

Arreglos

16 de septiembre

Ejercicio 3.1¶

Para cada una de las consignas siguientes, genere **un algoritmo** que permita solucionarla (3 algoritmos) usando un arreglo de 100 números enteros:

1. Almacenar 100 números.
2. Localizar el número de mayor valor y el de menor valor, informar sus valores y sus posiciones.
3. Contar y sumar todos los números pares.

definición:

```
[nombre_arreglo]: Arreglo de [tamaño (1…x)] de [tipo_de_dato]
```

Cómo encarar el ejercicio: Los arreglos tienen una cantidad fija y conocida de celdas. Si necesito cargar algo *iterativamente* una **cantidad fija de veces…***que estructura uso?* → *PARA*

Utiliza un contador entero, por lo que lo puedo usar para hacer referencia a los **índices del arreglo.** Un arreglo de 1 a 100 tiene índices de 1 a 100. por lo que “Para i:= 1 a 100” **i** en cada iteración puede hacer referencia a cada celda → arr[i]

En el punto 2 y 3 el arreglo ya está cargado, como indicamos eso en el algoritmo? pasa como ***parametro de la ACCION***

```
ACCION ejercicio3 (Arr: arreglo de [1..100] de enteros) ES
```

Punto 2: Aplicar la idea que ya conocemos para encontrar maximo y minimo en un conjunto de numeros

Punto 3: que operador podríamos usar para controlar fácilmente si un número es par? teniendo en cuenta que son aquellos divisibles por 2. → `num MOD 2 = 0`

1. Almacenar 100 números.

```
ACCION EJ31_A ES

AMBIENTE

    Ar: arreglo de [1..100] de entero

    //declarar indice de para!!

    num,i: entero

PROCESO

    //coincide con indices del arreglo:

    Para i:= 1 a 100 hacer
        ESC("Ingrese numero")
        LEER(num)
        Ar[i] := num
    finpara

FINACCION
```

2. Localizar el número de mayor valor y el de menor valor, informar sus valores y sus posiciones.Contar y sumar todos los números pares.

```
//2. Localizar el número de mayor valor y el de menor valor,

//informar sus valores y sus posiciones.

ACCION ej1_parte2(ar: arreglo de [1..100] de entero) es

AMBIENTE//suponemos el arreglo ya totalmente cargado

    i, mayor, menor, pos_mayor, pos_menor: entero

PROCESO

//el indice del para me ayuda a llevar control de en que celda estoy

    Para i:=1 a 100 hacer
        Si (ar[i] > mayor) entonces
            mayor:= vector[i]
            pos_mayor:= i
```

```
        fin_si
        Si (ar[i] < menor) entonces
            menor:= vector[i]
            pos_menor:= i
        fin_si
    fin_para

    Escribir("el entero más grande del vector es el:" ,mayor, "y está en la
posición",pos_mayor,)


    Escribir("el entero más pequeño del vector es el:" ,menor, "y está en la
posición",pos_menor,)

Fin_Accion
```

3. Contar y sumar todos los números pares.

```
//3. Contar y sumar todos los números pares.

ACCION ej31_c (ar: arreglo de [1..100] de entero) es

AMBIENTE//suponemos el arreglo ya totalmente cargado

    i, contador, acumulador: entero

PROCESO

contador:= 0

acumulador:= 0

    Para i:=1 a 100 hacer

        si (ar[i] MOD 2 = 0) entonces
            contador:=contador + 1
            acumulador:= acumulador + vector[i]
        fin_si

    fin_para

    ESC("La cantidad de numeros pares es: ", contador, "la suma de los numeros es:
", acumulador)

Fin_Accion
```

Ejercicio 3.2¶

Genere un único algoritmo que resuelva las 3 consignas del ejercicio anterior.

Cómo encarar el ejercicio: *Puedo realizar los 3 puntos en el mismo ciclo?* SI, podria ir cargando y analizando cada número, total va a terminar comparando **todos**

Podría ser un ciclo de carga del arreglo y otro ciclo que recorre y analiza punto 2 y 3, pero en este caso es posible hacerlo en conjunto

```
ACCION EJ32 ES

AMBIENTE

    Ar: arreglo de [1..100] de entero

    num,i,max,min,pos_max,pos_min,contador,acumulador: entero

PROCESO

    min := HV
    max := LV
    contador:= 0
    acumulador := 0

    Para i:= 1 a 100 hacer

        ESC("Ingrese numero")
        LEER(num)

        ar[i] := num

        //punto 2:

        Si (ar[i] > max) entonces
            max:= vector[i]
            pos_max:= i
        fin_si
        Si (ar[i] < min) entonces
```

```
            min:= vector[i]
            pos_min:= i
        fin_si
        //punto 3:
        si (ar[i] MOD 2 = 0) entonces
            contador:=contador + 1
            acumulador:= acumulador + vector[i]
        fin_si

    finpara

    Escribir("el entero más grande del vector es el:" ,max, "y está en la
posición",pos_max,)
    Escribir("el entero más pequeño del vector es el:" ,min, "y está en la
posición",pos_min,)
    Escribir("La cantidad de numeros pares es: ", contador, "la suma de los numeros
es: ", acumulador)

FINACCION
```

Ejercicio 3.4¶

Considerando un arreglo de 50 números enteros, confeccione un algoritmo para resolver las siguientes consignas:

1. Modificar el arreglo dado, de modo que todos sus elementos sean múltiplos de 3.
2. Crear otro arreglo que contenga los números que no cumplieron la condición.
3. Informar cuántos números cumplieron la condición.

Cómo encarar el ejercicio: *"Considerando un arreglo…" ya viene cargado! por lo que no lo podemos declarar en el ambiente, ¿qué hacemos? → pasa como* ***parametro de la ACCION***

Como no se el tamaño del arreglo de los que no cumple la condición, solo puedo saber que como **máximo** 50 números no pasan la condición (todo el arreglo). así que solo puedo declararlo de ese tamaño, *me puede sobrar pero no faltar..*

"Informar cuántos números cumplieron la condición." debería hacerlo cuando ya recorri todo el arreglo entonces! *luego de que termine el para*

```
ACCION 3.4 es (vector: arreglo de [1..50] de enteros) es
 AMBIENTE
    i, x: entero
    no_cumple: arreglo [1..50] de entero
 PROCESO
    x:=0
    Para i:=1 a 50 hacer
        Si (vector[i] MOD 3) <> 0 entonces
            no_cumple[x]:=vector[i]
            x:=x+1
            vector[i]:=vector[i]*3
        fin_si
    fin_para
    cantidad:= 50 - x
    Escribir("La cantidad de numeros que cumplieron la condición es de",cantidad,)
Fin_Accion
```

Ejercicio 3.5¶

Dados 2 vectores:

`A: arreglo [1 .. 30] de reales B: arreglo [1 .. 30] de reales`

Ambos ordenados de forma creciente, escribir un algoritmo que realice la mezcla de ambos para obtener otro vector también ordenado de forma creciente

`C: arreglo [1 .. 60] de reales`

```
ACCIÓN 3.5 (A,B: Arreglo [1...30] de Reales) es
AMBIENTE
    C:arreglo[1...60] de Reales
PROCESO
    PARA i:=1 a 60 HACER
        SI A[i] < B[e] ENTONCES
            C[j]:=A[i]
        SINO
            C[j]:=B[e]
            e:=e+1
        FIN_SI
```

```
    fin_para
FIN_ACCION
```

Hay un caso donde esta solución no funciona, ¿podes darte cuenta cuando?

la mejor manera es pensarlo como la **mezcla** de archivos, te lo dejo en C y te toca el pseudocódigo :)

```c
#include <stdio.h>
//EJERCICIO 3.4
int main() {
    // Declaración de los arreglos A, B y C
    float A[30];
    float B[30];
    float C[60];

    // Llenar los arreglos A y B (supongamos valores ordenados de forma creciente)
    // Aquí deberías llenar los arreglos A y B con los valores adecuados.

    // Índices para recorrer los arreglos A, B y C
    int i = 0;  // Para el arreglo A
    int j = 0;  // Para el arreglo B
    int k = 0;  // Para el arreglo C

    // Mezclar los arreglos A y B en C
    while (i < 30 && j < 30) {
        if (A[i] < B[j]) {
            C[k] = A[i];
            i++;
        } else {
            C[k] = B[j];
            j++;
        }
        k++;
    }

    // Si quedan elementos en A, copiarlos a C
    while (i < 30) {
        C[k] = A[i];
        i++;
        k++;
    }

    // Si quedan elementos en B, copiarlos a C
    while (j < 30) {
        C[k] = B[j];
        j++;
        k++;
    }

    // Imprimir el arreglo C resultante (ordenado)
    printf("Arreglo C resultante (ordenado de forma creciente):\n");
    for (k = 0; k < 60; k++) {
        printf("%.2f ", C[k]);
```

```
    }
    printf("\n");

    return 0;
}
```

Usa ciclo incluyente o excluyente? podría hacerse con cualquiera de los dos?

Ejercicio 3.9¶ (tarea clase anterior)

Se posee un arreglo de 200 libros con el siguiente formato:

AUTOR	NRO_LIBRO	TÍTULO	CANT_HOJAS

ordenado por **AUTOR** y se presentan las siguientes premisas:

1. Se necesita saber qué libros se poseen de “Nicklaus Wirth”.
2. Se necesita saber en qué posición se encuentra “Algoritmos + Estructuras de Datos=Programa”.
3. Se necesita saber cual es el libro de “Nicklaus Wirth” de mayor volumen.

```
ACCION 3_9 es (vector: arreglo [1...100] de libros) es
AMBIENTE
    libros = registro
        autor:AN(10)
        nro_libro: N(3)
        titulo: AN(20)
        cant_hojas: N(3)
    fin_registro
    mayor_vol: entero
    título_mayor: alfanumerico
PROCESO
    mas_grande:=0
    PARA i:= 1 a 100 HACER
        SI vector[i].autor = "Nicklaus Wirth" ENTONCES
```

```
                ESC("Se posee el siguiente titulo del autor Nicklaus Wirth: ",
vector[i].titulo)
            SI vector[i].cant_hojas > mayor_vol ENTONCES
                mayor_vol:= vector[i].cant_hojas
                título_mayor:= vector[i].titulo
            FIN_SI

                SI vector[i].titulo = "Algoritmos + Estructuras de Datos=Programa"
ENTONCES
                    ESC("Algoritmos + Estructuras de datos = Programa esta en la
posicion: ", i)
            FIN_SI
        FIN_SI
    FIN_PARA
    ESC("El título con el volumen más grande tiene el título",titulo_mayor)
FIN ACCION
```

¡Ahora tenemos que manejar más de 1 índice! [fila,columna]

¡A practicar cómo funcionaban los **paras anidados**!

¿Qué devuelve esto?

```
Para i:= 1 a 3 hacer

    ESC("El indice i es: ", i)

    Para j:= 1 a 2 hacer

        ESC("El indice j es: ", j)

    fin para

finpara
```

salida

El indice i es 1

El indice j es 1

El indice j es 2

El indice i es 2

```
El indice j es 1
El indice j es 2
El indice i es 3
El indice j es 1
El indice j es 2
```

Comenzamos matrices! 🙋

Ejercicios guia realizados en lenguaje C

Ejercicio 3.4¶

3.4 Hecho en “C”

```
#include <stdio.h>
#include <stdlib.h>  // Include for rand() and srand()
#include <time.h>    // Include for time()

// Function to display numbers in the "no_multiplos_de_3" array
void displayNoMultiplosDe3(int arr[], int size) {
    printf("Números que no son múltiplos de 3:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arreglo[50];
    int no_multiplos_de_3[50];
    int i, j = 0, k = 0;

    // Seed the random number generator
    srand(time(NULL));

    // Inicializar el arreglo con valores aleatorios (puedes cambiar esto según tus
necesidades)
    for (i = 0; i < 50; i++) {
        arreglo[i] = rand() % 100;  // Genera números aleatorios entre 0 y 99
    }
```

```
    // Modificar el arreglo para que todos los elementos sean múltiplos de 3
    for (i = 0; i < 50; i++) {
        if (arreglo[i] % 3 != 0) {
            no_multiplos_de_3[k] = arreglo[i];
            k++;
            arreglo[i] = (arreglo[i]*3);  // Hacer que sea múltiplo de 3
        } else {
            j++;
        }
    }

    // Informar cuántos números cumplieron la condición
    printf("Números originales que son múltiplos de 3: %d\n", j);
    printf("Números originales que no son múltiplos de 3: %d\n", k);

    // Llamar a la función para mostrar los números no múltiplos de 3
    displayNoMultiplosDe3(no_multiplos_de_3, k);

    return 0;
}
```

Pueden descargar el compilador de C y agregarlo a algún editor de texto (VSCode, sublime, etc) o sin necesidad de descargar nada, usar una **ide online:**

https://www.programiz.com/c-programming/online-compiler/

otras ideas que pueden probar en C para practicar manejo de vectores y matrices:

2. Write the code to fill the array `square` as shown:

1				
	4			
		9		
			16	
				25

3. Write the code to fill the array `square` as shown:

				5
			10	
		20		
	40			
80				

5. The partially initialized array "table" can be viewed as a primitive spreadsheet, in which the last column and bottom row have been left blank. Write the code to fill in this row and column with the totals of each column, each row, and the grand total.

1	2	3	4	5	
2	4	6	8	10	
20	10	5	3	1	
3	6	9	12	15	

una ayudita, asi lo podes declarar:

```
int table[ 5 ] [ 6 ] = { { 1, 2, 3, 4, 5 },
                         { 2, 4, 6, 8, 10 },
                         { 20, 10, 5, 3, 1 },
```

```
{ 3, 6, 9, 12, 15 } };
```