Universidad de La Frontera
Departamento de Ingeniería de Sistemas

CAPITULO 6: ARREGLOS

PROFESOR MAURICIO DIEGUEZ | PROGRAMACION DE COMPUTADORES | MODULO 4 | 23/09/2013

Usamos un arreglo para procesar una colección de datos del mismo tipo, como una lista de temperaturas registradas durante un día o una lista de nombres obtenida desde un archivo de alumnos de un curso. En este capítulo te presentaremos los fundamentos de la definición y uso de arreglos en los lenguajes C, C++ y Java, y muchas de las técnicas básicas que se emplean para diseñar algoritmos y programas que usan arreglos.

NOMBRE DEL ALUMNO: _____

Índice

Tema: 6.2 Arreglos Unidimensionales.

Subtema: 6.2.4. Utilizando Ciclos para Acceder a un Arreglo.

Dificultad: Fácil - Resueltos.

Ejercicio 1: Crear vector entero de 10 casillas, rellenar con los números del 1 al 10 y mostrar por pantalla.

Análisis: Simplemente, se trata de declarar e inicializar un vector del tipo entero con los valores del 1 al 10 y mostrarlo por pantalla. El vector puede ser inicializado durante la declaración o a través de un ciclo, sin embargo, para mostrarlo si necesitaremos un ciclo.

Entradas: No hay entradas.

Procesos:

-Declarar e inicializar vector.

-Iniciar ciclo.

-Mostrar elemento del vector.

-Finalizar ciclo.

Salidas:

-Vector – Vector entero array – 10 casillas.

Ejercicio 2: Definir e inicializar un vector del tipo entero sin el uso del teclado. Obtener y mostrar por consola tanto el mayor como el menor de los números. La cantidad de casillas del vector es ingresada por teclado.

Análisis: Para inicializar el vector, será necesario un ciclo para recorrer el arreglo y rellenar elemento a elemento con números aleatorios o también se podría inicializar al declarar el vector, esto último, sin la utilización de ciclos. En este caso, se inicializará con números aleatorios con la ayuda de la clase “Random” del paquete “útil”. Además, se deberá encontrar el mayor y el menor, para lo cual se utilizarán dos variables que se irán comparando con los valores dentro del vector. Dichas variables, se deberán inicializar con el primer valor ingresado, para así partir del supuesto que el primer valor es el mayor y el menor e ir comparando desde ese punto.

Nota: La importación, declaración e inicialización para la clase que nos permitirá utilizar números aleatorios sería como sigue:

```
1 Import java.util.Random;

2 Clase{
```


Volver

```
3       Método main{
4               Random aleatorio = new Random();
5               Int numeroAleatorio = aleatorio.nextInt(n);
        }
  }
```

En la línea 4, se declara e inicializa nuestra variable que nos abastecerá de métodos para obtener números aleatorios.

En la línea 5, por medio del método "nextInt" de nuestro objeto Random, llamado "aleatorio", se le asigna a la variable entera "numeroAleatorio" un número aleatorio, que puede ir desde 0 hasta n-1, donde n es el número que se utilice como argumento (lo que va dentro del paréntesis).

Entradas:

-Cantidad de casillas – Variable **entera** **casillas** – Mayor a 0.

Procesos:

-Importar clases.

-Declarar variable lectora.

-Declarar variable Random.

-Declarar vector.

-Iniciar ciclo.

-Añadir número aleatorio al vector.

-Finalizar ciclo.

-Inicializar variables max y min.

-Iniciar ciclo.

-Buscar máximo y mínimo.

-Finalizar ciclo.

-Mostrar máximo y mínimo.

Salidas:

-Máximo – Variable **entera** **max**.

-Mínimo – Variable **entera** **min**.

Dificultad: Fácil - Propuestos.

Ejercicio 1: Crear dos vectores de 5 casillas cada uno, rellenar un vector con los nombres y el otro con los apellidos. Se deben desordenar los nombres y apellidos. Al final de la ejecución se deben mostrar los nuevos nombres con apellido.

Ejercicio 2: Crear vector entero de 5 espacios y rellenarlo con números aleatorios. El objetivo es atinarle a la posición en la que se encuentra el número de menor valor. Si le atinas a la posición en la que se encuentra el número de mayor valor, o al segundo más alto, pierdes. Al perder o ganar, mostrar mensaje con los números y sus posiciones, además de un mensaje deacuerdo al contexto.

Ejercicio 3: Crear un vector del tipo double de 5 casillas, rellenarlo por teclado y obtener el número mayor del vector. Mostrar por pantalla el número obtenido.

Dificultad: Mediano - Resueltos.

Ejercicio 1: Inicializar un vector con las letras del abecedario y luego mostrarlo por pantalla. Los caracteres son diferencias por escribirse entre comillas simples, a diferencia de los String que se escriben entre comillas dobles.

Análisis: Se debe inicializar el vector de caracteres dentro de la misma declaración. Luego se debe mostrar a través de un ciclo.

Entradas: No hay entradas.

Procesos:

-Declarar e inicializar vector.

-Iniciar ciclo.

-Mostrar elemento del vector.

-Finalizar ciclo.

Salidas:

-Vector – Vector char **abc –** 27 casillas.

Dificultad: Mediano - Propuestos.

Ejercicio 1: Programa que simule, de manera básica, un juego de apuestas en el cual hay 6 números que cuentan como aciertos y 6 oportunidades de acertar. Los números a probar suerte son obtenidos de forma aleatoria y no por teclado, mientras que los números ganadores son siempre los mismos. Al final del programa se debe mostrar la cantidad de aciertos que hubo.

Ejercicio 2: Realizar un programa para hallar coincidencias de personalidad dentro de la sala de clases. La cantidad de personas a participar es definida al inicio del programa.

Preguntas:

a)¿Cuál es tu color favorito?

1-Rojo

2-Azul

3-Negro

4-Morado

5-Blanco

6-Celeste

7-Café

8-Rosado

9-Verde claro

10-Verde oscuro

b)¿Si te encuentras un paquete cerrado en la calle..?

1-Lo abres

2-Lo entregas a la policia

3-Buscas al dueño tu mism@

c)¿Para tu cumpleaños tu prefieres..?

1-Dinero

2-Regalos sorpresa

3-Regalos escogidos por ti(dentro de lo que se puede pagar

d)¿Te considerarías apasionad@?

1-Mucho

2-Un poco..

3-Soy algo timid@

e)¿Crees en el romanticismo y la magia que conlleva?

1-Si

2-Talvéz

3-No

f)Imaginese de niño.. Dentro de una bolsa existen algunos juguetes. ¿Cuál escoge usted?

1-Autitos

2-Legos

3-Pelota

4-Figuras de acción

5-Muñecas

6-Cubo rubix

7-Pistolitas

8-Juego de tasitas de té

9-Dibujitos para colorear

Mostrar la primera persona ingresada + <3 + la persona con la que logró mas coincidencias en sus respuestas.

Volver

Dificultad: Complejo - Resueltos.

Ejercicio 1: Declare un vector String de 6 casillas que se llene por teclado. Dar opcion de trasladar un elemento desplazando a los demas, cambiar de posiciones, mostrar vector y mostrar elemento dada la posición.

Análisis: Lo primero es declarar el vector, para luego rellenarlo a través de un ciclo con números ingresados por teclado. Debe aparecer entonces, un menú con todas las opciones requeridas.

Para trasladar un elemento, desplazando los demás, se debe tener en cuenta que si el elemento que queremos trasladar está a la derecha del espacio al que lo queremos trasladar, entonces los elementos que estaban entre medio se desplazarán hacia la derecha. En caso contrario, cuando el elemento a trasladar está a la izquierda del espacio al que queremos trasladarlo, los elementos entre medio serán desplazados hacia la izquierda. Además, es necesaria una variable auxiliar para alojar temporalmente el valor de la variable a trasladar. Para cambiar las posiciones también es necesario el tener una variable auxiliar.

Para el menú existirán 5 opciones: Trasladar y desplazar, intercambiar posiciones, mostrar vector, mostrar elemento y salir.

Entradas:

-Vector – Vector String **array** – 6 casillas.

-Opción – Variable **entera op** – Rango [1,5].

Procesos:

-Importar e inicializar lector.

-Declarar arreglo.

-Iniciar ciclo.

    -Ingresar dato en arreglo.

-Finalizar ciclo.

-Iniciar ciclo.

    -Mostrar menú.

    -Ingresar opción.

    -Realizar acción.

-Finalizar ciclo.

Salidas:

-Vector String **array** – 6 casillas.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Crear algoritmo que procese una fila de personas para ser tratadas en urgencia. El programa, debe ser capaz de priorizar según lo ordene el usuario, es decir, si el usuario decide que los ancianos son prioridad, los ancianos deben estar al comienzo de la fila. Las posibles prioridades son tres; los ancianos, los niños, enfermos graves.

El número de personas en la fila, el nombre de cada persona y, si es que se encuentra dentro de alguna de las posibles prioridades, la prioridad en la que se encuentra.

Al final del algoritmo, se deben mostrar los nombres, la prioridad y el número de lista.

Tema: 6.2 Arreglos Unidimensionales.

Subtema: 6.2.5. Búsquedas.

Dificultad: Fácil - Propuestos.

Ejercicio 1: Cree un algoritmo en el que se inicialize un vector de 10 espacios con números enteros desde el teclado. Facilite un sistema de búsqueda, en el que el usuario ingresa un número y el programa responde si ese número está o no dentro del vector. Si el número resulta estar en el vector el programa debe revelar su posición.

Ejercicio 2: Cree un vector del tipo String y rellénelo con 5 nombres. Se debe ingresar un nombre y el programa debe dar la posición del nombre en el arreglo, si es que este se encuentra en el arreglo, en caso contrario, avisar que este nombre no se encuentra en la lista.

Dificultad: Mediano - Propuestos.

Ejercicio 1: Inicialice un vector del tipo entero de 5 espacios con números aleatorios entre 1 y 10. El usuario tiene 3 oportunidades para ingresar un número y acertar. Mostrar mensaje cada vez que se ingrese un número indicando si acertó o no.

Ejercicio 2: Cree un vector con los sueldos finales de 5 empleados, teniendo en cuenta el pago por horas extras. El sueldo base es de $220000 y, sobre 40 horas de trabajo, el sueldo aumenta en $50000. Permitir ver que posiciones tienen un sueldo alto y cuales un sueldo bajo. Las horas trabajadas por cada empleado son ingresadas por teclado.

Dificultad: Complejo - Resueltos.

Ejercicio 1: La tia del quiosko le pide que le haga un programa en java para agilizar las ventas. Necesita saber el precio en todo momento de cualquier producto, elegir cuantas unidades se desean comprar, llevar la cuenta y el registro para mostrar al final en la tabla. Son 10 productos, cuyos nombres son ingresados por teclado, al igual que sus precios. Para comprar un producto, se debe seleccionar la opción “Buscador de precio” y luego, ingresar la cantidad por comprar.

Menú:

-Tabla con 10 productos y sus precios correspondientes

-Buscador de precio(Insertar nombre del producto)

-Boleta

Al final del programa se debe mostrar una tabla con los productos, la cantidad vendida de cada producto, ganancia por producto y utilidades totales.

Análisis: Debido a que se requiere que cada producto tenga un nombre para ser buscado, además de un precio y la cantidad que se ha vendido, suena mucho más conveniente el tener tres vectores. Un vector String, para los nombres y dos vectores int, para los precios y las cantidades.

El algoritmo debe ser capaz de, dado un nombre, encontrar el precio de un producto. Además, el algoritmo debe ser capaz de emitir una boleta en la que se mostrará una tabla con los productos, ganancias por producto, las cantidades vendidas y al final las utilidades totales obtenidas.

Para buscar el precio por el nombre, simplemente se debe ingresar el nombre y compararlo con cada elemento del arreglo String, hasta que se encuentre el que es de igual nombre. Al encontrarlo, se debe mostrar por pantalla el valor precio del producto. Luego de eso, preguntar si se desea comprar dicho producto y, en caso de que la respuesta sea afirmativa, preguntar la cantidad a comprar y acumular el número ingresado en el elemento del arreglo de cantidades correspondiente. Cada vez que se compre, también se debe acumular la ganancia para todos los productos, la cual nos dará las utilidades al final del algoritmo.

En la boleta, no se utilizarán más variables para las ganancias por producto, sino que multiplicara el valor de la cantidad por el producto dentro de la misma salida.

Entradas:

-Lista nombre productos – Vector String **nombres** – 10 casillas.

-Lista precios – Vector **entero** **precios** – 10 casillas – Elementos Mayores a 0.

-Lista cantidades – Vector **entero** **cantidades** – 10 casillas – Elementos Mayores a 0.

-Opción menú – Variable **entera** **op** – Rango [1,4].

Procesos:

-Importar e inicializar lector.

-Declarar vectores.

-Iniciar ciclo.

-Ingresar producto.

-Ingresar precio.

-Inicializar elemento vector cantidades.

-Finalizar ciclo.

Volver

-Inicializar acumulador utilidades.

-Iniciar ciclo.

-Mostrar menú.

-Ingresar opción.

-Realizar acción.

-Acumular cantidades.

-Acumular utilidades.

-Finalizar ciclo.

-Mostrar boleta.

Salidas:

-Vector String **nombres**.

-Cantidad de productos – Vector **entero** **cantidades**.

-Utilidades – Variable **entera** **útil**.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Luego de aprobar programación en la UFRO, usted es seleccionado para probar y despachar sistemas operativos por una prestigiosa empresa del rubro. Existen distintos sistemas operativos, cada uno con su ID, su propio valor y descuentos aplicados según cantidad. Además, según la cantidad a comprar, cambia el costo del despacho.

ID	Precio
FLX-2000	$5790
LR100XF	$8325
X86N99	$4450

Cantidad	Coste Despacho
De 1 a 9	No hay despacho
De 10 a 50	$10000
De 51 a 200	$15000
Más de 200	$20000

Cantidad	Descuento
1 a 5	5%
6 a 20	9%
Más de 20	15%

Se desean hacer indefinidas compras y por cada compra se debe mostrar el monto de la compra. Al final de la ejecución , se debe mostrar la ganancia total y el número de despachos, además de la opción de, ingresado un ID, mostrar la cantidad vendida y el dinero total descontado durante las ventas del producto de dicha ID.

Tema: 6.2 Arreglos Unidimensionales.

Subtema: 6.2.5. Ordenamiento.

Dificultad: Fácil - Resueltos.

Ejercicio 1: Ordenar un vector del tipo decimal, rellenado con números ingresados por teclado, de forma ascendente y mostrarlo por consola.

Análisis: Lo primero que se debe notar es que no se nos indica la cantidad de elementos que albergará el arreglo, por lo tanto eso lo deberá decidir el usuario. El momento de ingreso de la cantidad de casillas debe ser obviamente antes de declarar el vector y después de importar y declarar nuestro lector. El ingreso de la cantidad de casillas debe ser validado, esto para evitar el ingreso de números menores a 1.

Se ingresan los datos uno a uno a través de un ciclo y se guardan en el vector de tipo decimal. Más tarde, se debe utilizar el método burbuja, el cual simplemente compara el primer valor con el segundo valor y, si este es mayor, se intercambian y así sucesivamente cosa que el mayor siempre va quedando al final, proceso que repite casillas veces, con el propósito del orden ascendente. Con el fin de intercambiar las variables se utilizará una variable auxiliar, que servirá para almacenar el valor de uno de los elementos de manera temporal. Para acabar, sólo basta otro ciclo para mostrar el arreglo por pantalla.

Entradas:

-Cantidad de casillas del vector – Variable **entera** **casillas** – Mayor a 0.

-Vector – Vector **decimal** **array** – "casillas" casillas.

Procesos:

-Importar y declarar lector.

-Ingresar cantidad de casillas.

-Declarar vector.

-Iniciar ciclo.

    -Ingresar elementos.

-Finalizar ciclo.

-Iniciar ciclo.

    -Iniciar ciclo.

        -Comparar elementos dentro del arreglo.

        -Intercambiar valores.

-Finalizar ciclo.

-Finalizar ciclo.

-Iniciar ciclo.

-Mostrar elemento.

-Finalizar ciclo.

Salidas:

-Vector **decimal** **array**.

Dificultad: Fácil - Propuestos.

Ejercicio 1: Inicializar un vector del tipo entero con números enteros aleatorios del 1 al 200 y ordenarlo de mayor a menor. Mostrar dicho vector por pantalla. El vector debe tener un tamaño tan grande como el usuario lo requiera.

Dificultad: Mediano - Propuestos.

Ejercicio 1: Crear un vector decimal de 7 espacios y rellenarlo por teclado sólo con numeros positivos. Mostrar el vector ordenado por burbuja, de menor a mayor, luego invertir y volver a mostrar.

Ejercicio 2: Ingresar las notas de 5 alumnos, siendo 3 notas por alumno. Agrupar los promedios en un vector y mostrarlos ordenados de manera ascendente y descendentemente.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Debe crear un programa para las listas de urgencias especiales. Para eso usted cuenta con una lista de posibles accidentes vergonsosos, un nivel de importancia y un nivel de estabilidad(es por paciente). Debe ordenar la lista de ingreso a urgencia según puntaje de urgencia, contando con lo siguiente:

Puntaje de urgencia = Nivel de importancia / Nivel de estabilidad

A continuación se muestran los niveles de importancia:

lvl 12 ** Golpeado por una pequeña de 5 años.

lvl 60 ** Lengua pegada a un poste durante epoca invernal.

lvl 40 ** Miedo al ayudante de programación.

lvl 200 ** Trauma por abstinencia al juego League Of Legends.

Mínimo 7 pacientes. La estabilidad se ingresa por teclado y va de 1 a 10. El motivo de la urgencia del paciente también es ingresado por teclado y va de 1 a 4. Mientras mayor sea el número de estabilidad mejor será el estado del paciente, es decir, más atrás estará en la cola. Al final, mostrar la lista ordenada de pacientes con su puntaje de urgencia y el accidente que le ha ocurrido.

Ejercicio 2: Termina siendo un ayudante geek de programación. Entonces el profesor a cargo, aprovechandose de tus bastas habilidades y experiencia (la verdad solo le dió flojera), te pide que le hagas un programa para obtener los promedios de sus alumnos. Se ingresa el nombre del alumno, el promedio talleres, promedio tareas, prueba 1 y prueba 2.

Talleres: 15%

Tareas: 5%

Prueba 1: 40%

Prueba 2: 40%

Debe mostrar una tabla con nombre, promedio y el número de lista. Además se debe dar la opción de colocar una persona de una posición hacia cualquier otra desplazando los demás. También dar la opción de ordenar ascendente o descendentemente según promedio.

Nota: el número de alumnos es ingresado por teclado.

Ejercicio 3: Programar un algoritmo para que realice la tarea de revisar pruebas. Son 10 alumnos y 5 preguntas por prueba. Se deben ingresar las respuestas a medida que el programa las pida. Al final se debe mostrar la nota de cada alumno y el número de matrícula de cada alumno. Se debe dar la opción de buscar las notas de un alumno ingresando su número de matrícula.

Las preguntas y respuestas deben ser diseñadas por el programador. Para poder evaluar la prueba, debe tener en cuenta que se comienza con punto base 2 y que cada pregunta valdrá 1 punto.

Tema: 6.3 Arreglos Multidimensionales.

Subtema: 6.3.4 Acceso a una Matriz.

Volver

Dificultad: Fácil - Resueltos.

Ejercicio 1: Generar una matriz decimal de 4x5, rellenarla con números aleatorios, mostrarla y luego mostrar la traspuesta.

Análisis: Primero que nada veamos cómo se declaran las matrices.

```
Double nombre_matriz [][] = new Double [4][5];
```

Explicación: La palabra reservada "Double" indica el tipo de datos que albergará esta matriz. Luego, "nombre_matriz" puede ser cualquier nombre, por el cual quieran referirse a la matriz dentro del código. El primer doble corchete indica que el arreglo es de dos dimensiones, es decir, que tiene filas y columnas. Dentro del segundo doble corchete, va la cantidad de filas y columnas que tendrá nuestra matriz respectivamente.

Para asignar un valor a un elemento de la matriz, es parecido a como se hacía con los vectores.

```
nombre_matriz[n][m] = 15;
```

En el espacio ubicado en la fila n y la columna m, se le asigna a la matriz el valor 15.

Mostrar una matriz, requiere de la utilización de dos ciclos, uno para recorrer las filas y el otro para recorrer las columnas. El ciclo que recorre las columnas, se debe encontrar dentro del ciclo que recorre las filas, para así pasar por todos los elementos de la matriz. Así, por cada ciclo de columnas, se muestra el elemento actual de la matriz y por cada ciclo de filas se baja una línea, de este modo, se mostrará la matriz ordenadamente.

Para proseguir, el mostrar la matriz de manera transpuesta es más simple de lo que parece. Obtener la matriz transpuesta es simplemente mostrar cada fila hacia abajo y cada columna hacia la derecha. Dicho de otra manera, en vez de por cada ciclo "grande" o "externo" se mostrará una columna y por cada ciclo "pequeño" o "interno" se mostrará una fila.

Entradas: no hay entradas.

Procesos:

-Importar, declarar e inicializar clase Random.

-Declarar matriz.

-Inicializar contador fila.

-Iniciar ciclo filas.

-Inicializar contador columna.

-Iniciar ciclo columnas.

Volver

-Asignar valor al elemento.

-Mostrar elemento.

-Actualizar contador columna.

-Finalizar ciclo columnas.

-Saltar de línea.

-Actualizar contador fila.

-Finalizar ciclo filas.

-Inicializar contador columna. ←Comienza el proceso para mostrar la matriz transpuesta.

-Iniciar ciclo columnas.

-Inicializar contador fila.

-Iniciar ciclo filas.

-Mostrar elemento matriz.

-Actualizar contador fila.

-Finalizar ciclo filas.

-Salto de línea.

-Actualizar contador columna.

-Finalizar ciclo columnas.

Salidas:

-Matriz – Matriz decimal **cuadrado**.

Ejercicio 2: Se requiere de un algoritmo simple, el cual lleve a cabo la tarea de inicializar y mostrar una matriz del tipo entera. La inicialización se debe realizar por teclado y la matriz debe contener tantas filas y columnas como el usuario desee.

Análisis: Como es obvio, antes de declarar la matriz se deben ingresar la cantidad de filas y columnas. Luego de eso, simplemente se usará el doble ciclo para recorrer tanto filas y columnas dos veces; la primera vez, para poder ingresar los datos y la segunda para poder mostrar los datos.

Entradas:

-Cantidad de filas – Variable **entera** **fil** – Mayor a 0.

-Cantidad de columnas – Variable **entera** **col** – Mayor a 0.

-Matriz – Matriz **entera** **datos** – fil*col casillas.

Volver

Procesos:

-Importar, declarar e inicializar lector.

-Ingresar cantidad de filas.

-Ingresar cantidad de columnas.

-Declarar matriz.

-Inicializar contador fila.

-Iniciar ciclo filas.

    -Inicializar contador columna.

-Iniciar ciclo columnas.

    -Ingresar dato matriz.

    -Actualizar contador columna.

-Finalizar ciclo columnas.

-Actualizar contador fila.

-Finalizar ciclo filas.

-Inicializar contador fila.

-Iniciar ciclo filas.

    -Inicializar contador columna.

-Iniciar ciclo columnas.

    -Mostrar dato matriz.

    -Actualizar contador columna.

-Finalizar ciclo columnas.

-Saltar línea.

-Actualizar contador fila.

-Finalizar ciclo filas.

Salidas:

-Matriz **entera** **datos** – fil*col casillas.

Volver

Dificultad: Fácil - Propuestos.

Ejercicio 1: Dada una matriz decimal de NxM, donde N y M son números ingresados por teclado, rellenarla con valores aleatorios. Debe permitir obtener el valor en una posicion (x,y) y mostrarle por pantalla.

Ejercicio 2: Sumar 2 matrices enteras de 3x3 término a término y mostrar por pantalla. Las matrices son inicializadas con números aleatorios.

Dificultad: Mediano - Propuestos.

Ejercicio 1: Dadas dos matrices decimales A y B, intercambiar los mínimos de A con los máximos de B. Dichas matrices se inicializan por teclado. El tamaño de las matrices debe ser ingresado por teclado y debe ser igual para ambas matrices.

Nota: Se deben considerar 2 mínimos y 2 máximos.

Ejercicio 2: Dada una matriz 5x6 del tipo entera, intercambiar los elementos de la primera columna con la última columna. Dicha matriz debe ser inicializada por teclado.

Ejercicio 3: Ingresar monto en ventas diario de 10 vendedores durante 3 dias. Mostrar el promedio de ventas por día por cada vendedor y el promedio de ventas por dia entre todos.

Ejercicio 4: Debe diseñar un programa que lleve un registro del trafico de libros pertenecientes a la biblioteca. Se debe ingresar el día en que se retira, el día en que se entrega y el nombre del alumno. Según el número de dias que se ha tardado en devolver el libro , se debe enviar un mensaje por pantalla aplicando las reglas de la biblioteca. El número de alumnos entregando libros es ingresado por el usuario. Al final del algoritmo, se debe mostrar una tabla con los nombres, dias en que sacaron libros, dias en que se devolvieron y dias de castigo.

Ej: El alumno Gabriel se ha tardado 3 dias en devolver el libro.

Mensaje: Gabriel tiene un castigo de 3 dias de suspensión.

Nota: Asuma que son 2 días hábiles para devolver el libro. Por cada día de retraso, se contarán 3 días de castigo en el que el sospechoso no podrá pedir libros.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Contar el número de dígitos de cada elemento de una matriz 3x6 del tipo entera y mostrarlo junto a su posición. La matriz debe ser rellenada con números ingresados por teclado.

Ejercicio 2: En una conversación usted menciona que sabe programar en java, a lo que el dueño de un estacionamiento lo escucha y le ofrece una peguita. Debe diseñar un programa que pueda mantener registro sobre los estacionamientos ocupados y los desocupados, asi como el registro del tiempo y el precio a pagar por dicho tiempo. El tiempo ocupado por cada vehiculo es aleatorio. El estacionamiento cuenta con 12 espacios. El precio está a $350 la media hora.

Menu:

1-Mostrar espacios (1 = ocupado) (0 = desocupado)

[0] [0] [0] [1] [1] [0]

[1] [0] [1] [1] [0] [1]

2-Ingresar Auto

3-Retirar auto(Mostrar boleta con tiempo, posición y costo)

Ejercicio 3: Buscar la siguiente secuencia en una matriz, ya sea, horizontal como verticalmente.

1 1+2 1+2+3 1+2+3+4 1+2+3+4+5...

Se deben buscar 4 términos por lo menos. La matriz es de 10x10, rellenada por números aleatorios. Al final mostrar la matriz y el número de veces que se ha encontrado la sucesión dentro de la matriz.

Ejercicio 4: Programa que genera el cuadrado mágico. El programa debe colocar los números del 1 al 9 en una matriz de 3x3 de forma que la suma de cada fila, de cada columna y de las diagonales tenga el mismo resultado.

Tema: 6.3 Arreglos Multidimensionales.

Subtema: 6.3.4 Acceso a una Matriz - Búsquedas.

Dificultad: Fácil - Resueltos.

Ejercicio 1: Dadas dos matrices de diferentes tamaños, R y S, las cuales se inicializan por teclado con números enteros, mostrar los elementos comunes que tengan R y S. El tamaño de cada matriz es ingresado por teclado, siendo el tamaño igual al número de filas e igual al número de columnas.

Análisis: Nótese que los tamaños no son mencionados, por tanto, se deberán ingresar por teclado. Los datos también deben ser ingresados por teclado, sin embargo, como son matrices de diferentes tamaños se deberán inicializar dentro de ciclos dobles distintos.

Para poder comparar cada valor de la matriz R con todos los valores de la matriz S, simplemente se utilizaran 4 ciclos anidados, uno dentro de otro. Los dos primeros ciclos determinaran la posición del elemento de la matriz R, mientras que los otros dos ciclos más internos determinaran la posición del elemento de la matriz S. Haciendo esto, podremos comparar cada elemento de la matriz R con todos los elementos de la matriz S, y todo esto independiente de la diferencia en sus tamaños.

Entradas:

-Cantidad de filas matriz – Variable **entera** **fil1** – Mayor a 0.

-Cantidad de columnas matriz R – Variable **entera** **col1** – Mayor a 0.

-Cantidad de filas matriz S – Variable **entera** **fil2** – Mayor a 0.

-Cantidad de columnas matriz S – Variable **entera** **col2** – Mayor a 0.

-Matriz R – Matriz **entera** **R** – fil1*col1 casillas.

-Matriz S – Matriz **entera** **S** – fil2*col2 casillas.

Procesos:

-Importar, declarar e inicializar lector.

-Ingresar cantidad filas matriz R.

-Ingresar cantidad columnas matriz R.

-Ingresar cantidad filas matriz S.

-Ingresar cantidad columnas matriz S.

-Declarar matrices.

-Rellenar matrices desde el teclado.

-Iniciar ciclo fil1.

-Iniciar ciclo col1.

-Iniciar ciclo fil2.

-Iniciar ciclo col2.

-Comparar elementos.

-Mostrar elemento común.

-Finalizar ciclo col2.

-Finalizar ciclo fil2.

-Finalizar ciclo col1.

-Finalizar ciclo fil1.

Salidas:

-Elemento común – Variable **entera común**.

Dificultad: Fácil - Propuestos.

Ejercicio 1: Cree una matriz de 3x6 y rellene con números aleatorios enteros. El usuario ingresa una posición (x, y) y el programa debe devolver la cantidad de números del mismo valor que se encuentren en su vecindad.

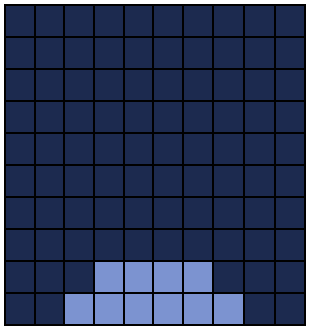
Dificultad: Mediano - Propuestos.

Ejercicio 1: Cree e inicialice una matriz de MxN, siendo N y M ingresados por teclado, con números enteros aleatorios entre 1 y N*M. Al ingresar un número se debe mostrar por consola la cantidad de veces que ese número se repite dentro de la matriz.

Ejercicio 2: Crear un programa que sirva como base de datos. En dicho programa se debe ingresar la matrícula, el nombre y las notas. Se debe realizar el algoritmo para 5 alumnos con 3 notas cada alumno. Al final mostrar una tabla con el número de matrícula, el nombre, las notas y el promedio, además de la opción de buscar una matrícula y que aparezcan los datos correspondientes a dicha matrícula, además de si pasó o no de curso.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Usted debe encargarse de los tickets a la venta para una película en el cine, siendo la mejor opción el crear un programa. Los asientos normales cuestan $2500, mientras que los vip cuestan $5000. Debe mostrar la matriz y dar tres opciones; Ingresar un asiento a comprar; comprobar si un asiento está ocupado; salir, mostrar las ganancias obtenidas y la cantidad de ventas vip. Debe ser una matriz de 10x10, en la cual existen 10 asientos vip (azules).



Ejercicio 2: Crear un programa que simule el juego de la batalla naval, contando para ello, con matrices. Para cada jugador debe haber 5 barcos; uno de 3 espacios de ancho y 5 de alto, dos de 2 espacios de ancho y 1 de alto, dos de 3 espacios de ancho y 1 alto. Las matrices deben ser de 10x10. Cada vez que se le atina a un barco enemigo, debe aparecer un mensaje informando de la situación. Para eliminar un barco se le debe atinar a cada espacio que le compone. Al final mostrar un mensaje con el ganador del juego.

Nota: Son los jugadores quienes deben escoger las posiciones de los barcos, basta con dar la posición extrema izquierda del barco para que el programa sepa cuales otras posiciones se ocuparán.

Ejercicio 3: Realizar un programa que simule un juego de buscaminas. Para ello debe tener en cuenta lo siguiente:

-Será en un espacio de 10x10

-Habrán 9 bombas escondidas aleatoriamente.

-Al seleccionar una coordenada, se debe mostrar la cantidad de bombas alrededor de dicha coordenada. En caso de haber seleccionado una bomba, perder el juego.

-El juego se acaba cuando no quedan más coordenadas sin bombas a las que apuntar o cuando se apunta a una bomba.

Tema: 6.3 Arreglos Multidimensionales.

Subtema: 6.3.4 Acceso a una Matriz - Ordenamiento.

Dificultad: Fácil - Resueltos.

Ejercicio 1: Inicializar una matriz de 4x3 con números con decimales por teclado. Ordenar cada fila de menor a mayor y mostrar por consola

Análisis: Para realizar este ejercicio, simplemente se debe tener en mente a cada fila de la matriz como si este fuese un vector, de este modo, será más fácil entender cómo utilizar el método burbuja para todas las filas.

Lo primero, es inicializar la matriz con el doble ciclo por teclado. Luego, para poder ordenar cada fila, se debe hacer un ciclo para recorrer cada fila de la matriz, tal como se acostumbra, y luego, ordenar dicha fila como si fuese un vector. Para esto último, se utilizará el método burbuja.

Entradas:

-Matriz – Matriz decimal números – 4*3 casillas.

Procesos:

-Importar y declarar lector.

-Declarar matriz.

-Ingresar datos matriz.

-Iniciar ciclo filas.

-Iniciar ciclo.

-Iniciar ciclo.

-Comparar valores en la fila.

-Intercambiar valores.

-Finalizar ciclo.

-Finalizar ciclo.

-Finalizar ciclo filas.

-Mostrar matriz.

Salidas:

-Matriz decimal numeros – 4*3 casillas.

Volver

Dificultad: Fácil - Propuestos.

Ejercicio 1: Guardar en una matriz los números de lista y los promedios de 5 alumnos. Ordenar los promedios de forma descendente y que sigan concordando con sus números de lista. Mostrar lista.

Dificultad: Mediano - Propuestos.

Ejercicio 1: Existen 5 productos a la venta en su sección:

ID	Precio
005423	$5215
001634	$2105
003498	$1640
004169	$4165
003247	$7990

Crear un algoritmo para obtener los datos de un día de compras. Para ello hacer uso de matrices (ID x Precio). Al final del algoritmo debe mostrar una tabla con las ID y las ganancias de su respectivo producto, dando las opciones de ordenar tanto ascendente como descendentemente (con respecto a la ganancia por producto) y de buscar, a través de la ID, la ganancia correspondiente al producto.

Está en la libertad del usuario decidir cuándo se acaba el día de venta.

Ejercicio 2: Se realiza una competencia de obstáculos entre 3 equipos de 4 personas. Se deben ingresar, para 10 competencias, las personas que compiten y los tiempos registrados. Cada competencia tiene el mismo puntaje. Cada persona puede participar como máximo tres veces y como mínimo una vez. Las personas dentro del grupo son consideradas como números del 1 al 4.

Al final del algoritmo, se deben mostrar en una tabla a todos los competidores con su grupo, su mejor tiempo y su peor tiempo. La tabla debe estar ordenada de menor a mayor con respecto a los mejores tiempos. Además, mostrar el equipo vencedor y su número de victorias.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Se debe crear un algoritmo para clasificar 3 pelotones según su puntería. Cada pelotón consta de 5 soldados.

En una prueba, a campo abierto, los soldados ponen a prueba sus habilidades mientras compiten con los otros pelotones. Por soldado, se ingresan la cantidad de tiros acertados, la cantidad de tiros realizados y el pelotón al que pertenece.

$$\text{Puntería} = \frac{Número\ de\ tiros\ acertados}{Número\ de\ tiros\ realizados}$$

Se debe mostrar una tabla con los puntajes de puntería; se deben mostrar los soldados del mejor pelotón, ordenados de mejor puntería a peor puntería; luego, los soldados del segundo mejor pelotón, ordenados de mejor puntería a peor puntería; por último, los soldados del peor grupo, ordenados de mejor puntería a peor puntería.

Puntería	Clasificación
Menor a 0.2	Useless
Mayor o igual a 0.2 y menor a 0.4	Support
Mayor o igual a 0.4 y menor a 0.7	Assault Force
Mayor o igual a 0.7 y menor a 1	Sniper
Igual a 1	Master of Death

En la tabla, deben mostrarse los soldados (soldado 1, soldado 2, etc…), su puntería, el pelotón al que pertenece y su calificación militar. Además, debe mencionarse que pelotón fue el que obtuvo mejores resultados.

Ejercicio 2: A usted, camino a su trabajo en el servicio técnico para computadoras, se le ocurre la idea de un algoritmo para simplificar su trabajo. El algoritmo debe recibir la ID de la computadora y el problema a solucionar. Además, debe ingresarse el nivel de gravedad y el nivel de urgencia. La cantidad de computadoras a reparar es ingresada al inicio.

Gravedad	Nivel
Muerto	4
Muy mal	3
Reparable	2
Fácil de reparar	1

Urgencia	Nivel
Importante	3
Puede esperar un poco	2
El tiempo es irrelevante	1

Coste de trabajo: 10000*nivel de gravedad*nivel de gravedad*nivel de urgencia.

Se debe trabajar con matrices alfanuméricas y enteras. Al final, mostrar las ID's con el problema, la gravedad de la situación, la urgencia de la situación y el costo por el trabajo. Mostrar la opción de ordenar ascendente o descendentemente según la gravedad, precio o urgencia.

Tema: 6.6 Ejercicios Propuestos – Ejercicios mezclados Areglos.

Dificultad: Fácil - Resueltos.

Ejercicio 1: Determinar que columna de la matriz N x M es igual al vector de tamaño N. Tanto la matriz como el vector son rellenados por teclado con números enteros. El tamaño de la matriz es ingresado por teclado.

Análisis: Es fácil notar que se dice indirectamente en el ejercicio que las dimensiones de los arreglos deben ser ingresadas por el usuario y que la cantidad de casillas del vector debe ser igual a la cantidad de filas de la matriz. Esto significa que solo se deben ingresar la cantidad de filas y columnas de la matriz, ya que la cantidad de filas de la matriz será igual a la cantidad de casillas del vector.

Para saber que columna de la matriz es igual al vector, se debe comparar cada columna de la matriz con el vector haciéndolo término a término. Esto se realizará pensando en que, tanto columna como vector, deben tener igual valor en igual posición, de esta forma será un poco más simple. Lo que se hará, será que se utilizará un contador extra, para así, poder contar la cantidad de datos iguales y si este contador llega a ser igual al número de filas, entonces serán iguales.

Nota: Arreglos es una forma general de llamarles. Tanto la matriz como el vector son arreglos, la diferencia está en que la matriz tiene dos dimensiones y el vector una.

Entradas:

-Cantidad de filas – Variable **entera** **fil** – Mayor a 0.

-Cantidad de columnas – Variable **entera** **col** – Mayor a 0.

-Matriz – Matriz **entera** **cuadrado** – fil*col casillas.

-Vector – Vector **entero** **línea** – fil casillas.

Procesos:

-Importar, declarar e inicializar clase lectora.

-Ingresar cantidad filas.

-Ingresar cantidad columnas.

-Declarar matriz y vector.

-Ingresar datos matriz.

-Ingresar datos vector.

-Inicializar contador extra..

-Iniciar ciclo.

-Iniciar ciclo.

-Comparar valores en la posición.

-Actualizar contador extra.

-Finalizar ciclo.

-Mostrar posición columna igual a vector.

-Reiniciar contador extra.

-Finalizar ciclo.

Salidas:

-Contador de columnas – Variable entera i.

Dificultad: Fácil - Propuestos.

Ejercicio 1: Transformar una matriz del tipo entero en un vector pasando cada dato de la matriz hacia el vector. La matriz es inicializada por teclado y su tamaño también es ingresado por el usuario. Al final del algoritmo se debe mostrar el vector con los datos.

Dificultad: Mediano - Propuestos.

Ejercicio 1: Se debe crear un subprograma para mostrar los puntajes de un mini-juego del estilo shooter online. Se deben ingresar la cantidad de puntajes a mostrar, los nombres, los puntajes y el número de asesinatos por persona. El algoritmo, debe ser capaz de mostrar una tabla con los nombres, los puntajes y las muertes, ordenada de mayor a menor según los puntajes.

Ejercicio 2: Se ha observado que quien deja de entrenar, aunque sea un día, comienza a perder su nivel de habilidad y su condición física. Se ingresan al estudio 5 personas con sus nombres, sus años de entrenamiento y la cantidad de días que han faltado al entrenamiento. Las personas se clasifican según su nivel de habilidad en el deporte.

Nivel de habilidad = ValorEntero$*\left(\frac{Años\ de\ entrenamiento}{0.62*Días\ faltados}\right)$

Mostrar el nombre, los años de entrenamiento y el porcentaje de días sin entrenar de la persona con mayor nivel de habilidad y de la persona con menor nivel de habilidad.

Dificultad: Complejo - Propuestos.

Ejercicio 1: Dada una matriz Z del tipo entero, almacenar en un vector A la suma de todos los elementos de cada columna y en un vector B la suma de todos los elementos de cada fila. La matriz debe inicializarse por teclado. Al final mostrar los vectores A y B, los valores idénticos entre Z, A y B y el valor máximo entre A y B.

Ejercicio 2: Has logrado ahorrar mucho dinero, por lo que al fin te decides a comprar un TV led. Para elegir un TV de acuerdo a tus recursos y necesidades, se tienen en cuenta las siguientes prioridades.

Nivel de prioridad	Resolución
37	HD
51	Full HD

Nivel de prioridad	Tipo de contraste
62	Estático
33	Dinámico

Nivel de prioridad	Frecuencia de imagen
20	50-60 Hz
70	100-120 Hz

Nivel de prioridad	Tamaño
50	32
60	39
62	40
48	42

Precio del TV led = Valor aleatorio entre 180000 y 400000.

Nivel de prioridad del TV = sumatoria de los niveles de prioridad.

Generar 10 TV's led con características y precios aleatorios almacenando toda la información. El usuario es quien define el rango de precios que le es accesible. El algoritmo debe mostrar una tabla con las TV's dentro del presupuesto, con todas sus características, incluyendo el precio. Esta tabla, debe estar ordenada según el interés, desde la TV con mayor nivel de prioridad hasta la de menor nivel.

Análisis: Básicamente, se debe realizar la creación de una matriz entera y una matriz String. La matriz entera se encargará de guardar los precios y los niveles de prioridad (total por tv), mientras que la matriz String se encargará de guardar las características de cada televisor. Para rellenar las matrices, se utilizará la clase Random, obteniendo las características y niveles de prioridad de cada televisor.

La idea, es que a través de condicionales y números aleatorios podamos asignarle a cada televisor una característica aleatoria.

Se debe notar, que para obtener las características aleatorias de cada televisor, se debe recorrer un ciclo, el cual abarcará cada televisor y dentro de este ciclo, se obtendrá un número aleatorio, dependiendo de este la característica a agregar. Se deben obtener tantos números aleatorios como características hayan. Cada vez que se obtenga una característica aleatoria, se debe acumular el nivel de prioridad del caso en el elemento de la matriz entera, además de añadir la cualidad a la matriz String. Al final de la obtención de características para un televisor, se debe calcular el precio y almacenarlo en la matriz entera.

Llegados a este punto, es importante recordar que se deben entregar los resultados según el nivel de prioridad acumulado de cada televisor. Es por esto, que es importante ordenar las matrices según el nivel de prioridad acumulado para así mostrar estos datos de manera ordenada en la tabla al final del algoritmo.

Cabe sugerir, que al mostrar la tabla, se debe tener conciencia de que sólo se mostrarán los televisores dentro del rango de precios ingresado. Es por esto, que se debe hacer uso de un condicional dentro del ciclo para filtrar los televisores según el rango de precios.

Entradas:

-Precio mínimo – Variable **entera** **min** – Rango [180000, 400000].

-Precio máximo – Variable **entera** **max** – Rango [min, 400000].

Procesos:

-Importar, declarar e inicializar clase lectora.

-Importar, declarar e inicializar clase para números aleatorios.

-Declarar matrices.

-Inicializar matriz entera.

-Iniciar ciclo Tv's.

-Obtener resolución aleatoria.

-Actualizar matrices.

-Obtener contraste aleatorio.

-Actualizar matrices.

-Obtener frecuencia aleatoria.

-Actualizar matrices.

-Obtener tamaño aleatorio.

-Actualizar matrices.

-Obtener precio.

-Actualizar matriz entera.

-Finalizar ciclo.

-Ordenar matrices por burbuja.

-Ingresar precio mínimo.

-Ingresar precio máximo.

-Iniciar ciclo.

-Iniciar ciclo.

-Mostrar dato matriz en tabla.

-Finalizar ciclo.

-Mostrar dato vector en tabla.

-Finalizar ciclo.

Salidas:

-Matriz – Matriz **entera** **NivelxPrecio** – Elemento: Con precio en rango [min,max].

-Matriz – Matriz String **caracteristicas** – Elemento: Con precio en rango [min,max].

Volver