UTN Facultad Regional La Rioja

Trabajo Practico Nº 02

Funciones, Vectores, Matrices y Punteros

2024

Índice

Introducción	3
Entrega	3
Criterio de evaluación	3
Rubrica de Evaluación	4
Enunciados	5
Ejercicio 01:	5
Ejercicio 02:	5
Ejercicio 03:	5
Ejercicio 04:	6
Ejercicio 05:	6
Ejercicio 06:	6
Ejercicio 07	6
Ejercicio 08	6
Ejercicio 09	6
Ejercicio 10	7

Introducción

El presente trabajo práctico tiene como objetivo poner en práctica los conocimientos adquiridos en la materia hasta ahora.
Los ejercicios propuestos abarcan el uso de variables, funciones básicas como printf y scanf, así como el manejo de estructuras condicionales (if, if/else, switch) y estructuras cíclicas (while, do while, for), vectores, matrices, funciones, y punteros.

Es fundamental comprender y aplicar adecuadamente estos conceptos para resolver eficientemente los problemas planteados. Además, se espera que los estudiantes adquieran habilidades en la organización del código, la lectura de entradas desde el teclado, y la presentación de resultados en la terminal.

Entrega

Se deberá entregar un archivo comprimido con winrar en formato .zip o .rar, este archivo deberá contener:

1. Un archivo PDF con los códigos realizados por cada ejercicio junto con la imagen de la salida de la Terminal (Aplicación Programada).
2. Un archivo de código fuente por cada ejercicio realizado, en este archivo debe estar el código funcional y comentado correctamente.

Nombres de los Archivos:
Los archivos deberán tener el siguiente formato de nombre:

- **Archivo Comprimido**: "TP 02 - Comisión X - Apellido Nombre"
  - *Ejemplo*:
    - TP 02 – Comisión B – Oviedo Carlos.rar
- **Archivo PDF**: “TP 02 - Comisión X - Apellido Nombre ".
  - *Ejemplo*:
    - TP 02 – Comisión B – Oviedo Carlos.pdf
- **Código Fuente:** " TP 02 - Comisión X - Apellido Nombre – Ejercicio xx"
  - *Ejemplo***:**
    - TP 02 – Comisión B – Oviedo Carlos – Ejercicio 01.c

Fecha de Entrega: *domingo 09 de junio del 2024 a las 23:59 hs*

Criterio de evaluación

Los trabajos prácticos de programación se calificarán según una rúbrica de 10 puntos por ejercicio, luego para la nota final del práctico se dividirá el puntaje total en 10. Criterios a evaluar:

- **Funcionamiento del programa**: Cada programa debería funcionar correctamente en todas las entradas. Además, si hay alguna especificación sobre cómo funciona el programa debería escribirse, o cómo debe aparecer el resultado, se deben seguir esas especificaciones.
- **Legibilidad**: Las variables y funciones deben tener nombres significativos. El código debe organizarse en funciones cuando corresponda. Debe haber una cantidad adecuada de espacio en blanco para que el código sea legible y la sangría debe ser coherente.
- **Documentación**: El código y funciones deben estar comentados apropiadamente. Sin embargo, no todas las líneas deben ser comentadas porque eso hace que el código esté demasiado ocupado. Pensar detenidamente dónde se necesitan comentarios.
- **Elegancia del código**: Hay muchas formas de escribir la misma funcionalidad en el código y algunas de ellas son innecesariamente lentas o complicadas. Por ejemplo, si está repitiendo el mismo código, debería estar dentro de una nueva función o bucle for.
- **Especificaciones y entrega del práctico**: Cada programa debe ser guardado con un determinado nombre de archivo u otras especificaciones similares y debe ser entregado en tiempo y forma. Ver especificaciones de entregas indicadas anteriormente.

Rubrica de Evaluación

Funcionamiento	*4*	*3*	*2*	*0*
	El programa siempre funciona correctamente y cumple con las especificaciones.	El programa no cumple con detalles menores de las especificaciones, o funciona incorrectamente en algunas entradas.	El programa no cumple con detalles significativos de las especificaciones, o exhibe un comportamiento incorrecto.	El programa no funciona, o no compila.
Legibilidad	*2*	*1*	*0,5*	*0*
	El código es limpio, entendible y bien organizado.	Problemas menores como sangría inconsistente, denominación de variables, organización general	Al menos un problema importante que dificulta la lectura.	Varios problemas importantes que dificultan su lectura.
Documentación	*2*	*1*	*0,5*	*0*
	El código está bien documentado.	Unos o dos lugares pueden beneficiarse de comentarios, o el código está comentado de más	La falta de comentarios dificulta la compresión del código.	El código no presenta comentarios.
Elegancia		*1*	*0,5*	*0*
		El código utiliza adecuadamente bucles for y métodos para código repetido, y la codificación es mínima.	El código utiliza un enfoque mal elegido al menos en un lugar, por ejemplo, codificar algo que podría implementarse a través de un bucle for.	En varias ocasiones en que el código podría haber utilizado un enfoque más fácil/rápido/mejor.
Especificaciones		*1*	*0,5*	*0*
		El ejercicio cumple con las especificaciones.	El ejercicio no cumple con especificaciones menores.	El ejercicio no cumple con especificaciones significativas.
Entrega			*0*	*-3*
			Se entregó dentro del plazo establecido.	Se entregó fuera del plazo establecido.

Enunciados

Ejercicio 01:

Escribe un programa en C que muestre un menú, el cual deberá tener estas opciones:
Cada opción del menú y hasta el propio menú debe estar en un procedimiento (o función) diferente.

1. Cargar Dimensión del Vector. (Puede ser de 2 a 9999)
2. Generar Vector (generar un vector con números aleatorios)
3. Calcular el valor promedio.
4. Mostrar valores máximo y mínimo y su posición correspondiente.
5. Ordenar de mayor a menor.
6. Ordenar de menor a mayor.
7. Buscar un valor determinado [*Usar búsqueda binaria.]*
   (pedir ingresar un valor para luego buscarlo e indicar su posición en caso de encontrarlo)
8. Salir

Las variables de las opciones 3, 4, 5 deberán estar en la función principal y trabajar con punteros para modificar su valor.

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 02:

Realice un programa en C que almacene en vectores el nombre, la edad y la estatura de un grupo de n personas y que, a través de funciones, permita buscar la posición del vector en que se encuentra una persona cualquiera, buscada por su nombre y la edad que esta persona tiene, el promedio de edad y el promedio de estatura.

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 03:

Codificar una función que acepte dos cadenas de caracteres como parámetros, y que compruebe si la segunda cadena s2 es una subcadena de la primera si. El programa devolverá un entero que indicará la posición del primer carácter de la segunda cadena s2 dentro de sí. Devolverá -1 si s2 no está contenida en sí. Si existen varias ocurrencias de s2 en sí se devolverá la posición de la primera de ellas, ignorando el resto.
La cabecera (o prototipo) de la función será:

- int buscar (char sl[ ], char s2[ ]);

Ejemplo:

1. si = "Imagina que hay una GUERRA, y no vamos NADIE"
   s2 = "que"

   La función devolverá el entero 8.

2. si = "las lágrimas te impedirán ver las estrellas -Tagore-"
   s2 = "im"

   La función también devolverá el entero 8
(porque es la posición de la primera ocurrencia dé las dos que hay).

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 04:

Escribe un programa en C que solicite al usuario ingresar el tamaño n de un vector (el tamaño deberá estar definido entre qué valores debe elegir el usuario) y luego los n números enteros que compondrán el vector. Implementa el algoritmo de ordenamiento por burbuja para ordenar el vector en orden ascendente. Después de ordenar el vector, elimina los elementos duplicados y muestra el vector resultante sin duplicados.

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 05:

Escribe un programa en C, que el usuario ingrese una palabra u oración (máximo 100 caracteres) y que se indique si la misma es palíndromo.

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 06:

Escribe un programa en que solicite al usuario un año y con una función nos diga si un año es bisiesto o no. La función aceptará un único argumento que será el año que queremos saber si es o no bisiesto y devolverá un valor verdadero si es bisiesto y falso si no lo es.

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 07

Escribe un programa en C que permita almacenar números enteros entre 50 y 100 en una matriz de n filas y m columnas. Luego buscar un número cualquiera dentro del rango ingresado e indicar en qué posición (fila y columna) se encuentra, si es que está almacenado, de lo contrario, el programa deberá decir que el número buscado no está en la matriz.
Si el número se encuentra almacenado varias veces, el algoritmo retorna la posición de la primera ocurrencia.
NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 08

Crea un programa que calcule el determinante de una matriz cuadrada ingresada por el usuario. Puedes usar cualquier método de cálculo de determinante que prefieras.
(por ejemplo, el método de Laplace o la regla de Sarrus para matrices 3x3).

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 09

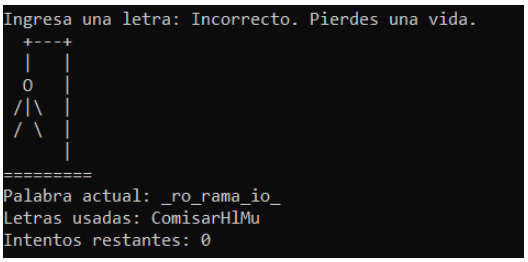
Escribe un programa que tome una matriz como entrada y calcule su matriz traspuesta.
El usuario deberá indicar el número de filas, columnas y los valores de cada elemento de la matriz.

NOTA: Validar todos los ingresos de datos del usuario.

Ejercicio 10

El programa selecciona aleatoriamente una palabra de una lista predefinida y le pide al usuario que adivine la palabra letra por letra. El usuario tiene un número limitado de intentos para adivinar la palabra completa.

Lista de Palabras:
"gato",
"perro",
"casa",
"ordenador",
"programación"



El Juego deberá ir indicando:

- Los intentos restantes.
- Las letras usadas.
- La palabra actual ocultando las letras con"_".
- Se deberá ir generando el dibujo del ahorcado.

En la imagen se observa un ejemplo de cómo se deberá ir dibujando y mostrando la información al usuario.

NOTA: Validar todos los ingresos de datos del usuario.