Informática
Guía de Trabajos Prácticos
Ingeniería en Mecatrónica

Unidad 1
1. Instalar GNU/Linux, bien en forma nativa o bien sobre una máquina virtual (ej: Virtual Box)
2. Instalar Eclipse para desarrollo en C/C++ (complemento CDT)
3. Crear un nuevo proyecto C en Eclipse

Unidad 2
4. Crear un programa en C que incluya
    1. El uso de las directivas del preprocesador
        1. #include
        2. #define
        3. #ifdef
        4. #endif
    2. Declaración de variables char, int, long, float
    3. Asignación de valores a estas variables
    4. Impresión de valores de las variables por pantalla con al función de la biblioteca estándar de entrada/salida (stdlibc) llamada printf

5. Compilar y ejecutar el programa. Ejecutar en modo depuración: insertar puntos de corte; ejecutar el programa paso por paso; inspeccionar las variables

6. Escribir un programa que lea desde el teclado 5 números enteros **a**, **b**, **c**, **d** y **e** y que imprima por pantalla el rango en el que se encuentra el número **e**. Ej: “b >= e >= c”

7. Escribir un programa que calcule el factorial de un número y muestre el valor por pantalla

8. Escribir un programa que permita ingresar por teclado un valor inicial **a**, un valor final **b**, y un número **c**, y que cuente la cantidad de números divisibles por **c** que hay en el rango **[a,b]**.

9. Escribir un programa que permita ingresar una variable **i**, y que calcule la siguiente ecuación:

$$f(i)=\sum_{j=0}^{i} j$$

10. Escribir un programa que calcule la siguiente ecuación:

$$\sum_{i=0}^{100}\sum_{j=0}^{100}\frac{i+j}{i-j} \qquad \forall i,j \;/\; (i-j)\neq 0$$

11. Escribir un programa que calcule la siguiente ecuación:

$$\sum_{i=0}^{100}\sum_{j=0}^{100}\frac{i+j}{i-j} \qquad \forall i,j \;/\; i \; y \; j \; son \; números \; pares$$

12. Escribir un programa que calcule la siguiente ecuación:

$$\sum_{i=0}^{100}\sum_{j=0}^{100}\frac{i+j}{i-j} \qquad \forall i,j \;/\; i \;\; o \; j \; son \; números \; pares$$

13. Escribir un programa que lea un valor entero desde el teclado, y que muestre su configuración de bits por pantalla utilizando los operadores **<<** y **>>**.

14. Escribir un programa que lea 2 números enteros por teclado, y que calcule
    1. el número que contiene sólo los bits que son 1 en **ambos** números
    2. el número que contiene los bits que son 1 en **alguno** de los números
    3. el número que contiene los bits que son **distintos** en ambos números

Dr. Ing. Martín Marchetta (mmarchetta@fing.uncu.edu.ar)

15. Escriba una función que reciba como argumentos 3 coeficientes **a**, **b** y **c**, y un valor de **x**, y que evalúe el polinomio:

$$p(x) = ax^2 + bx + c$$

16. Escriba un procedimiento main que lea desde el teclado 3 valores en punto flotante **x1, x2** y un incremento **delta**, y que calcule p(x) utilizando la función escrita en el ejercicio 15, en el intervalo **[x1, x2]**, según el incremento **delta.** Muestre los valores calculados por pantalla.

17. Implemente una función que calcule el factorial de un número **mediante recursión** y devuelva dicho factorlal. Llame a esa función desde el procedimiento main y muestre el resultado por pantalla

18. Implemente **recursivamente** el cálculo de la sucesión de Fibonacci:

$$f_0 = 0$$
$$f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \quad \forall\, n \geq 2$$

19. Implemente un programa que: (a) declare una variable *v* de tipo int; (b) llame a una función

    *void duplicar(int *v)*

    pasándole como parámetro la dirección de memoria de la variable v (mediante &v); (c) la función debe multiplicar *v* por 2 y colocar el resultado en la misma dirección de memoria, de manera que el programa *main* "vea" el cambio de valor; (d) imprima por pantalla el valor de la variable *v* para verificar que el programa funciona correctamente.

20. Escriba una función que genere un arreglo de N elementos aleatorios **enteros** (leer N desde el teclado). Mostrar los N elementos generados por pantalla usando printf.

21. Escriba una función para recorrer el vector y mostrar por pantalla solo aquellos elementos **pares**.

22. Escriba una función para recorrer el vector y copiar en un nuevo vector aquellos elementos que sean impares. Luego recorrer el nuevo vector y mostrar los elementos. El tamaño del nuevo vector será igual a la cantidad de elementos menores a 5 del vector anterior (utilizar asignación de memoria dinámica).

23. Escriba una función para recorrer el vector, sumar el contenido de aquellos elementos cuyo sub-índice sea impar en una variable. Mostrar la variable al finalizar por pantalla

24. Escriba una función para recorrer el vector y guardar el promedio de los valores en una variable. Mostrar el promedio por pantalla.

25. Escriba una función para recorrer el vector y encontrar el menor de los elementos. Mostrar por pantalla el menor

26. Implementar el ejercicio 15, pero permitiendo ingresar el orden del polinomio por teclado (es decir, se debe poder evaluar un polinomio de orden arbitrario). Utilice un arreglo grande (ej: 100 elementos) para almacenar los coeficientes.

27. Escriba una función que lea una cadena de caracteres por teclado (mediante scanf), que luego invierta el orden de los caracteres en la cadena, y finalmente muestre la nueva cadena por pantalla.

28. Escriba una función que lea un archivo con el siguiente formato:

    # Mapa a ser recorrido (1=ocupado, 0=libre)
    1000011111
    1100010000
    0100010000

0100010000
0100010000
0100011000
0100001111
0100000001
0100000001
0111111110

#posiciones inicial y final
inicio=(2,3)
fin=(8,8)

El símbolo '#' indica que la línea es un comentario, y que debe ser ignorada. La primera parte del archivo debe leerse y colocarse en un arreglo de 2 dimensiones de tipo int. La última parte del archivo ("inicio" y "fin") debe leerse y guardarse en 4 variables llamadas inicioX, inicioY, finX y finY.

29. Implementar una función que genere 2 matrices de orden m x n y n x p respectivamente, y que calcule la multiplicación de las matrices (mostrar resultados por pantalla). Asuma valores constantes para m, n y p (no los lea por teclado, asignelos usted mismo de manera que la multiplicación sea posible).

30. Implementar una función que genere 2 vectores y calcule el producto escalar de los mismos

31. Implementar una función que genere 2 vectores. Esta función debe luego llamar a otra función que calcule el producto vectorial de los mismos. Esta última función debe recibir como parámetro los 2 vectores a multiplicar, más un tercer arreglo en donde se colocarán los resultados (recuerde que los arreglos siempre se pasan por referencia)

32. Implementar una función que genere una matriz con todos los elementos en 0 excepto aquellos para los que **i+j** sea **par** (para estos elementos generar un valor aleatorio con la función rand).

33. Implementar una función que calcule la transpuesta de la matriz del ej. 32.

34. Transformaciones espaciales: **escalado**. Leer un vector de 3 dimensiones (x, y z) por teclado, luego leer 3 factores de escala (sx, sy, sz), e implemente una función que escala el vector de entrada de acuerdo a los factores leidos, generando el vector (x', y', z'). Cálculo:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sx & 0 & 0 \\ 0 & 0 & sx & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

35. Transformaciones espaciales: **traslación**. Leer un vector de 3 dimensiones (x, y z) por teclado, luego leer 3 valores de tralsación (x0, y0, z0), e implemente una función que traslada el vector de entrada de acuerdo a los factores leidos. Cálculo:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

36. Transformaciones espaciales: **rotación**. Leer un vector de 3 dimensiones (x, y z) por teclado, luego leer 3 valores de tralsación (x0, y0, z0), e implemente una función que traslada el vector de entrada de acuerdo a los factores leidos. Cálculo:

Rotación $\alpha°$ alrededor de x

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotación $\beta°$ alrededor de y

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotación $\theta°$ alrededor de z

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

NOTA: Las 3 matrices pueden combinarse en una sola multiplicándolas previamente entre sí, y utilizando el resultado como transformación acumulada. Ej: si la rotación en x, y, z utilizan las matrices A, B y C respectivamente, puede calcularse T = A x B x C, y utilizar directamente T para rotar un vector en los 3 ejes al mismo tiempo.

Ejercicios con uso de tipos definidos por el usuario y memoria dinámica

37. Implementar el ejercicio 26 (evaluación de un polinomio de orden arbitrario), reservando memoria de manera dinámica para contener los coeficientes. Lea también un intervalo [x1, x2] y un incremento Δx y evalúe el polinomio para todos los valores del intervalo utilizando el incremento leído entre uno y otro. Utilice la función calloc.

38. Implementar el ejercicio 29 (multiplicación de matrices) reservando memoria dinámicamente. Lea los tamaños de las matrices por teclado y verifique que la multiplicación puede realizarse.

39. Implementar el ejercicio 28 utilizando estructuras de datos dinámicas (arreglos con tamaño definido en tiempo de ejecución para la matriz del mapa).
    Consideraciones:
    1. Utilice la función calloc()
    2. Recuerde que para arreglos de N dimensiones, la memoria debe reservarse con calloc() de manera independiente para cada dimensión

40. Implementar el ejercicio 31 (producto vectorial), pero en lugar de utilizar arreglos, utilice un tipo definido por el usuario (struct) que contenga 3 elementos (x, y, z). Reserve memoria dinámicamente para estas esctructuras, y páselas por referencia a la función que calcula el producto vectorial (reserve memoria dinámica para los 2 vectores a multiplicar, y también para el resultado).

Unidad 3

41. Implementar el ejercicio 38 (multiplicación de matrices dinámicas) utilizando multithreading. Cada thread debe producir una **fila** de la matriz resultado: si A es de mxn y B es de nxp, el resultado de A x B es una matriz C de mxp, por lo que se crearán m threads para la multiplicación. Imprima el resultado por pantalla.

42. Implementar el ejercicio 41 mediante multiprocessing. En este caso se creará un proceso por cada fila de la matriz resultado. La matriz debe generarse antes de crear los procesos hijos de manera que cada hijo tenga una copia de la matriz. Cada proceso hijo, al terminar su trabajo, debe enviar la fila resultante al padre a través de un pipe. Utilice semáforos al momento de enviar los datos al padre. Imprima el resultado por pantalla.

43. Conectar un mouse USB a la máquina, y utilizar la biblioteca HID API para
    1. Enumerar los dispositivos USB HID conectados
    2. Abrir el dispositivo que corresponde al mouse para leer
    3. Leer los bytes del mouse, mostrando los valores de los bytes relevantes por pantalla (crudos)
    4. Decodificar el protocolo del mouse, mostrando los valores procesados por pantalla (incluidos los botones y la ruedita). Simule el movimiento del puntero, mostrando **los valores de la posición** (x, y) en pixels en la que se encontraría el puntero del mouse si se estuviera graficando; configure un valor mínimo (x, y) en (0, 0) para la esquina superior izquierda de la pantalla, y defina un MAX_X y MAX_Y para simular la esquina inferior derecha de la pantalla (esto es un manejo interno de su aplicación, no del hardware del mouse).

       Recuerde ejecutar el programa como root, y seguir las instrucciones para configurar la biblioteca de soporte libusb

       Consejos:
       – Identifique los bytes relevantes
       – Recuerde que los bytes se leen del puerto USB como unsigned char (bytes con valores 0 a 255 sin signo), pero la semántica de algunos de estos bytes en realidad sí tiene signo (-128 a 127, por ejemplo para representar izquierda/derecha o arriba/abajo); identifique los bytes que tienen signo, y conviértalos a char (que tiene signo) antes de procesarlos.
       – Recuerde identificar los bytes de los botones y la ruedita del mouse.

44. Mediante sockets TCP/IP implemente el ejercicio 43 con 2 procesos: uno (el cliente) lee los valores del mouse y se lo envía al otro (servidor); el servidor muestra por pantalla la representación de los valores. Los procesos pueden correr en la misma máquina o en máquinas remotas (esto es opcional).

    Nota: En caso de correr en el mismo host, se pueden configurar los sockets del cliente y del servidor con la IP del host, o bien con la IP 127.0.0.1 que representa el host local (loopback).