Ejercicios de C

1. Estructura de un programa

Objetivos:

- Practicar con el editor
  - usar el editor emacs `emacs nombre.c`
- y el compilador
  - para compilar y enlazar: `gcc -Wall nombre.c`
  - el ejecutable se llama `a.out`

Descripción:

- Escribir, compilar y ejecutar el programa que escribe "hola" en la pantalla

2. Declaraciones

Objetivos:

- Practicar con declaraciones de variables y constantes

Descripción:

- Crear un programa que declare varias variables y constantes enteras y reales
- El programa debe hacer operaciones simples con estas variables (+,-,*,/).
- Al final, mostrar los resultados.

```
printf("texto %d",expr. entera);     /* C */
printf("texto %f",expr. real);       /* C */
```


3. Arrays

Objetivos:

- Practicar con arrays de números reales

Descripción:

- Crear tres variables de tipo array de tres números reales (vectores)
- Leer por teclado los valores de dos de los arrays:

```
scanf("%f",&var_real);          /* C */
```

- Calcular la suma de los dos vectores, almacenándola en el tercero
- Mostrar la suma por pantalla

4. Strings

Objetivos:

- Practicar con strings de longitud variable

Descripción:

- Declarar 2 strings de hasta 80 caracteres
- Leer en uno tu nombre y en otro el nombre de tu padre

  scanf("%s",var_str); /* C */
- Poner en pantalla el mensaje "El padre de xxx es yyy"

  printf("texto %s",exp_str); /* C */

5. Tipos enumerados

Objetivos:

- Practicar con los tipos enumerados

Descripción:

- Crear un tipo enumerado para los meses del año (`meses`)
- Crear un array cuyo índice sean los meses del año, capaz de almacenar el número de días de cada mes. Inicializarlo
- Poner en pantalla los días que tiene el mes que indique el usuario.

5. Tipos enumerados (cont.)

Para escribir meses en C:

- Crear un array de strings con los nombres de cada mes:

```
typedef char nombre_mes[12];
nombre_mes nombre[] ={"enero","febrero",...};
```

- Escribir la casilla correspondiente:

```
mes=marzo;
printf("el mes de %s",nombre[mes]);
```

Para leer meses en C:

- leer el nombre y luego comparar uno por uno con los elementos del array `nombre[]`
- o leer el número del mes

6. Registros o Estructuras

Objetivos:

- Practicar con datos almacenados en un array de registros o estructuras

Descripción:

- Crear un tipo de datos para almacenar los datos personales de un alumno (nombre, dirección, año de nacimiento)
- Crear un array de datos personales
- leer por teclado el número de la casilla del array a usar
- leer los datos de un alumno, en esa casilla del array (usar "`fgets(string,n,stdin)`" para nombre y dirección)
- mostrar estos mismos datos

7. Instrucción condicional

Objetivos:

- Practicar con la instrucción condicional

Descripción:

- Leer por teclado tres números reales
- Calcular el máximo de los tres
- Mostrarlo por pantalla

8. Instrucción condicional múltiple

Objetivos:

- Practicar con la instrucción condicional múltiple

Descripción:

- Leer una nota de un alumno (entero entre 0 y 10)
- Poner en pantalla:
  - suspenso: 0<=nota<=4
  - aprobado: 5<=nota<=6
  - notable 7: <=nota<=8
  - sobresaliente: 9<=nota<=10
  - error: nota<0 ó nota>10

9. Lazo indefinido

Objetivos:

- Practicar con el lazo indefinido

Descripción:

- Escribir un programa que lea números y los vaya sumando, hasta que el número introducido sea negativo

10. Lazo for

Objetivos:

- Practicar con el lazo for

Descripción:

- Leer un número entero: N
- Calcular la suma de los N primeros números enteros

11. Funciones

Objetivos:

- Practicar con funciones y el paso de parámetros

Descripción:

- Definir un tipo vector-3D como un array de 3 casillas reales
- Crear tres funciones; no usar variables globales:
  - leer por teclado las tres componentes de un vector
  - producto escalar de dos vectores: se retorna el valor
  - escribir en pantalla las tres componentes de un vector
- Crear un programa principal que permita calcular el producto escalar de dos vectores y muestre al final los dos vectores y el resultado

12. Modularidad

Objetivos:

- Crear un módulo de programa con interfaz separada del cuerpo

Descripción:

- Definir el tipo vector y sus operaciones del ejercicio anterior en un módulo con interfaz separada del cuerpo
- Añadir una operación de suma de vectores
- Rediseñar el programa principal para:
  - usar el nuevo módulo
  - permitir al usuario elegir la operación a realizar con los vectores (suma o producto escalar)

13. Tratamiento de errores

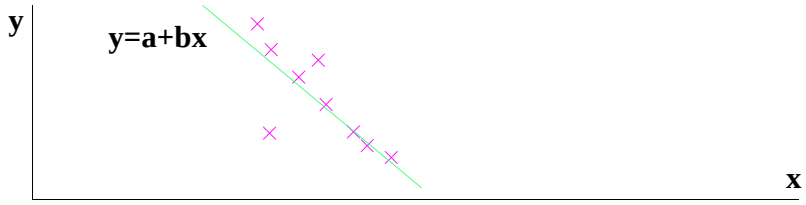
Objetivos:

- Practicar los métodos de tratamiento de errores aritméticos y de teclado

Descripción:

- Crear un programa para el cálculo de regresiones lineales
  - Preguntar el número de datos a introducir, N
  - Leer de teclado N parejas de datos (X,Y)
  - Mostrar los resultados:
    - constantes a y b de la recta
    - factor de correlación
- Tratar errores de teclado (preguntando de nuevo), así como errores aritméticos (división por cero)

Regresión Lineal



$$b = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2} \qquad a = \frac{\sum y - b\sum x}{n}$$

$$r^2 = \frac{(n\sum xy - \sum x\sum y)^2}{(n\sum x^2 - (\sum x)^2)\cdot(n\sum y^2 - (\sum y)^2)}$$