# Práctica 7 de programación en C

# Objetivos

- Aplicar las estructuras de selección y repetición a la resolución de problemas, incluyendo bucles anidados.
- Conocer la programación modular: funciones en C.
- Utilizar vectores unidimensionales en la resolución de problemas.

# Funciones en C

- Implementación de la función (se declara antes del programa principal (función **main**)):

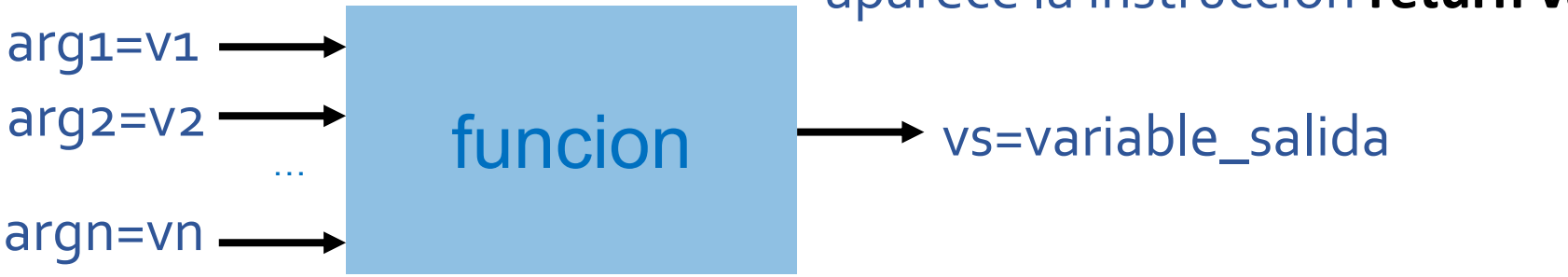
```
Tipo_variable_salida   funcion(tipo1 arg1, tipo2 arg2,..., tipon argn){
    ....
    return variable_salida
}
```

- Invocación desde otra función, en particular desde el programa principal:

```
....
vs=funcion(v1, v2, ....,vn);
....
```

vs=variable de salida (almacenará el resultado)
vi=variables o datos de entrada
Todas las variables deben ser del mismo tipo que las declaradas en la función

- Si en el **Tipo_variable_salida** se pone **void**, indica que no se devuelve ningún valor, y por lo tanto, no aparece la instrucción **return variable_salida**



# Ejemplo: cálculo del área de un rectángulo



```c
#include <stdio.h>
float  area_rectangulo(float base, float altura){
    float area;
    area=base*altura;
    return area;
}
int main(){
    float A, b, h;
    printf("Introduce la base del rectángulo: ");
    scanf("%f", &b);
    printf("Introduce la altura del rectángulo: ");
    scanf("%f", &h);
    A=area_rectangulo(b, h);
    printf("El área del rectángulo es igual a %f\n ",A);
    return 0;
}
```

# Ejercicio 4



- Desarrollar un programa en C que lea de un fichero información de una flota de aviones y calcule algunos datos estadísticos de los aviones.
- Se partirá de un fichero llamado **vuelos.txt** (se encuentra en Poliformat) con datos sobre los vuelos de diferentes aeronaves durante un determinado periodo de tiempo. Cada aeronave viene identificada por un número entero entre 0 y **N**-1, siendo **N** el número total de aeronaves de la flota.
- En el fichero mencionado se guardan tres valores enteros para cada vuelo realizado: identificador del avión (que se repetirá para todos los vuelos del mismo avión), distancia del vuelo (en kilómetros) y cantidad de combustible empleado (en litros). Para cada avión hay tantas líneas en el fichero (desordenadas) como viajes ha realizado. Los aviones que no han realizado ningún trayecto no aparecen en el fichero.

Ejemplo:

**41 6297 88159**

**62 850 14450**

**47 2965 47441**

La primera línea informa de un vuelo del avión con identificador 41 en el que ha recorrido 6297 kilómetros y para el que ha necesitado 88159 litros de combustible. La segunda línea indica que el avión 62 ha hecho un vuelo de 850 km gastando 14450 litros, ….

# Ejercicio 4

Hay que completar este código

ESCOLA TÈCNICA SUPERIOR ENGINYERIA

```c
#include <stdio.h>
#include <float.h>

#define N 82 // Número total de aeronaves


int menu() {
    int opc;

    /* muestra un menú al usuario y solicita una opción. Si la opción no
    está entre las del menú, indica un mensaje de error y vuelve a
    mostrar menú / pedir opción, repitiéndose todo hasta que se
    escoge una opción correcta */

    return opc;
}

int cargarDatos(int num_vuelos[], int distancias[], int consumos[]) {
    FILE *file;
    int id, distancia, consumo;

    file = fopen("vuelos.txt", "r");
    if (file == NULL)
        return 1;

    while (0 /*sustituir el 0 por la condición que corresponda -lectura de una línea-*/) {
        num_vuelos[id]++;
        distancias[id] += distancia;
        consumos[id] += consumo;
    }

    fclose(file);
    return 0;
}

void mostrarInformacionAvion(int num_vuelos[], int distancias[], int consumos[]) {
    int id;

    do {
        printf("Avion (0-%d): ", N-1);
        scanf("%d", &id);

        if (id < 0 || id >= N)
            printf("Error. Valor fuera de rango.\n");
    } while (id<0 || id >= N);

    if (num_vuelos[id] == 0) {
        printf("El avion %d no ha realizado vuelos.\n", id);
    } else {
        /* Falta el cálculo del consumo medio y mostrar por pantalla
        el número total de vuelos, la distancia total recorrida, el
        consumo total de combustible y el consumo medio por kilómetro. */
    }
}
```

```c
void mostrarEficienciaAviones(int num_vuelos[], int distancias[], int consumos[]) {
    int i, id_mas_eficiente = -1, id_menos_eficiente = -1;
    float consumo_medio_mas_eficiente = FLT_MAX, consumo_medio_menos_eficiente = 0, consumo_medio;

    for (i = 0; i < N; i++) {
        /* Falta completar el bloque del for: recorrido de los vectores
        para encontrar el avión más eficiente y el menos eficiente en
        términos de consumo medio por kilómetro */
    }

    if (id_menos_eficiente != -1) {
        printf("Avion menos eficiente: %d (consumo: %.1f l/km)\n", id_menos_eficiente, consumo_medio_menos_eficiente);
        printf("Avion mas eficiente: %d (consumo: %.1f l/km)\n", id_mas_eficiente, consumo_medio_mas_eficiente);
    }
    else
        printf("Ningún avión ha volado todavía.\n");
}

int main() {
    int num_vuelos[N] = {0};
    int distancias[N] = {0};
    int consumos[N] = {0};
    int i, opcion;

    if (cargarDatos(num_vuelos, distancias, consumos) == 1){
        printf("Error al abrir el fichero.\n");
        return 1;
    }

    do {
        opcion = 0/*cambiar el 0 por la llamada a la función menu */;

        switch(opcion) {
            case 1:
                mostrarInformacionAvion(num_vuelos, distancias, consumos);
                break;
            case 2:
                mostrarEficienciaAviones(num_vuelos, distancias, consumos);
                break;
            case 0:
                printf("Bye!\n");
                break;
        }
    } while(opcion != 0);

    return 0;
}
```

# Ejercicio 4



Al ejecutar el código, el programa mostrará por pantalla un menú con 3 opciones y solicitará al usuario que elija una de ellas. Tras comprobar la validez de la opción seleccionada, calculará los valores indicados por el usuario, los mostrará por pantalla y volverá a mostrar el menú por pantalla. El programa acabará cuando el usuario seleccione la opción 0.

- Opción 1: Información de un avión: Esta opción muestra datos estadísticos sobre los vuelos realizados por un avión en concreto.
- Opción 2: Aviones más y menos eficientes: Esta opción busca los aviones más y menos eficientes, entendiendo que un avión es más eficiente que otro si su consumo medio por kilómetro ha sido menor.

```
AVIONES
-------
1. Informacion de un avion.
2. Aviones mas y menos eficientes.
0. Salir.
Elige una opcion:
```

# Ejercicio 4 (función main)

1. Invoca a la función **cargarDatos** para almacenar en tres vectores la información contenida en los tres ficheros:
   - num_vuelos vuelos: almacena el número total de vuelos realizado por cada avión (pudiendo ser cero si aún no ha volado).
   - distancias: almacena la distancia total (en kilómetros) recorrida por cada avión.
   - consumos: almacena el consumo total (en litros) que ha tenido cada avión.
2. Dentro de un bucle do-while invoca a la función menú para obtener un número 0, 1 y 2. En función de la opción elegida ejecutará una u otra opción.

```
74  int main() {
75      int num_vuelos[N] = {0};
76      int distancias[N] = {0};
77      int consumos[N] = {0};
78      int i, opcion;
79
80      if (cargarDatos(num_vuelos, distancias, consumos) == 1){
81          printf("Error al abrir el fichero.\n");
82          return 1;
83      }
84
85      do {
86          opcion = 0 /*cambiar el 0 por la llamada a la función menu */;
87
88          switch(opcion) {
89              case 1:
90                  mostrarInformacionAvion(num_vuelos, distancias, consumos);
91                  break;
92              case 2:
93                  mostrarEficienciaAviones(num_vuelos, distancias, consumos);
94                  break;
95              case 0:
96                  printf("Bye!\n");
97                  break;
98          }
99      } while(opcion != 0);
100
101     return 0;
102 }
```

Declaración de variables

Invocación a **cargarDatos**

Se cambia una vez se haya implementado la función **menu**()

Obtención de un número 0, 1 o 2

**num_vuelos**: almacena el número total de vuelos realizado por cada avión (puede ser 0)
**distancias**: almacena la distancia total ( kilómetros) recorrida por cada avión.
**consumos**: almacena el consumo total ( litros) que ha tenido cada avión

# Desarrollo del ejercicio 4



- Completar la función cargarDatos: esta función almacena en tres vectores la siguiente información:
  - num_vuelos vuelos: almacena el número total de vuelos realizado por cada avión (pudiendo ser cero si aún no ha volado).
  - distanciasdistancias: almacena la distancia total (en kilómetros) recorrida por cada avión.
  - consumosconsumos: almacena el consumo total (en litros) que ha tenido cada avión.

```
18  int cargarDatos(int num_vuelos[], int distancias[], int consumos[]) {
19      FILE *file;
20      int id, distancia, consumo;
21
22      file = fopen("vuelos.txt", "r");
23      if (file == NULL)
24          return 1;
25
26      while (0 /*sustituir el 0 por la condición que corresponda -lectura de una línea-*/) {
27          num_vuelos[id]++;
28          distancias[id] += distancia;
29          consumos[id] += consumo;
30      }
31
32      fclose(file);
33      return 0;
34  }
```

```
41  6297  88159
62  850   14450
47  2965  47441
81  1053  13689
 1  3395  33950
 0  8705  156693
```

Nota: solo hay que cambiar la línea 26 por la lectura de  líneas del fichero **file** hasta alcanzar final de fichero

# Desarrollo del ejercicio 4



- Completar la función menú

```
int menu() {
    int opc;

    /* muestra un menú al usuario y solicita una opción. Si la opción no
    está entre las del menú, indica un mensaje de error y vuelve a
    mostrar menú / pedir opción, repitiéndose todo hasta que se
    escoge una opción correcta */

    return opc;
}
```

```
AVIONES
-------
1. Informacion de un avion.
2. Aviones mas y menos eficientes.
0. Salir.
Elige una opcion:
```

# Desarrollo del ejercicio 4



- Completar la función mostrarInformacionAvion

```
void mostrarInformacionAvion(int num_vuelos[], int distancias[], int consumos[]) {
    int id;

    do {
        printf("Avion (0-%d): ", N-1);
        scanf("%d", &id);

        if (id < 0 || id >= N)
            printf("Error. Valor fuera de rango.\n");
    } while (id<0 || id >= N);

    if (num_vuelos[id] == 0) {
        printf("El avion %d no ha realizado vuelos.\n", id);
    } else {
        /* Falta el cálculo del consumo medio y mostrar por pantalla
        el número total de vuelos, la distancia total recorrida, el
        consumo total de combustible y el consumo medio por kilómetro. */
    }
}
```

```
AVIONES
-------
1. Informacion de un avion.
2. Aviones mas y menos eficientes.
0. Salir.
Elige una opcion: 1

Avion (0-81): 82
Error. Valor fuera de rango.
Avion (0-81): 18
El avion 18 ha realizado 26 vuelos (75223 km) usando 1062236 l de combustible.
Consumo medio por Km: 14.1 l/km
```

- Para probar el correcto funcionamiento de esta opción, debéis de cambiar en la función **main**() la línea `opcion = 0;` por la línea `opcion = menu();`

# Desarrollo del ejercicio 4



- Completar la función mostrarEficienciaAviones

```
void mostrarEficienciaAviones(int num_vuelos[], int distancias[], int consumos[]) {
    int i, id_mas_eficiente = -1, id_menos_eficiente = -1;
    float consumo_medio_mas_eficiente = FLT_MAX, consumo_medio_menos_eficiente = 0, consumo_medio;

    for (i = 0; i < N; i++) {
        /* Falta completar el bloque del for: recorrido de los vectores
        para encontrar el avión más eficiente y el menos eficiente en
        términos de consumo medio por kilómetro */
    }

    if (id_menos_eficiente != -1) {
        printf("Avion menos eficiente: %d (consumo: %.1f l/km)\n", id_menos_eficiente, consumo_medio_menos_eficiente);
        printf("Avion mas eficiente: %d (consumo: %.1f l/km)\n", id_mas_eficiente, consumo_medio_mas_eficiente);
    }
    else
        printf("Ningún avión ha volado todavía.\n");
}
```

```
AVIONES
-------
1. Informacion de un avion.
2. Aviones mas y menos eficientes.
0. Salir.
Elige una opcion: 2

Avion menos eficiente: 46 (consumo: 15.5 l/km)
Avion mas eficiente: 4 (consumo: 13.0 l/km)
```

# Ejercicio 1



Desarrolla un programa con una única función (la función main) que pida un número entero al usuario (el número debe estar entre 1 y 10) y muestre por pantalla el factorial de dicho número.

Ejemplo de funcionamiento (en cursiva lo introducido por el usuario):

Dame un numero: *4*

El factorial de 4 es 24

Nota: n!=1*2*3*…*n

# Ejercicio 2



Desarrolla un programa que implemente dos funciones:

a. Una función, llamada factorial, que reciba un valor entero y devuelva el factorial de dicho valor (excepto si el valor recibido no es positivo, caso en el que devolverá el valor 0):

int factorial(int n)

b. Una función principal (main) que pida un número entero cualquiera y use la anterior función para calcular su factorial.

# Ejercicio 3

Desarrolla un programa que calcule y muestre:

$$1! + 2! + 3! + \ldots + (n-2)! + (n-1)! + n!$$

donde n es un valor que debe pedirse al usuario y tiene que estar entre 1 y 10. Utiliza en el programa la función factorial desarrollada en el ejercicio 2.

Ejemplo de funcionamiento:

```
Calculo de 1! + 2! + 3! + ... + (n-1)! + n!

Dame el valor de n: 0
Dame el valor de n: 22
Dame el valor de n: 5
El resultado es: 153
```

# Ejercicio 3



```
#include <stdio.h>

Incluir la función factorial del ejercicio 2

int main(){
    int n, f, total=0;
    printf("Calculo de 1! + 2! + 3! + ... + (n-1)! + n!\n\n");

    Leer un dato n por el teclado hasta que su valor se encuentre entre 1 y 10

    Bucle for que recorra los valores i=1,2,…,n
        total=total+factorial(i);

    printf("El resultado es: %d\n", total);
    return 0;
}
```

# Muchas gracias
# y
# ¡ a programar !