# *UF 1: PROGRAMACIÓN ESTRUCTURADA*

**EJERCICIO 1: (2.5 puntos) (cada error encontrado → 0,5 puntos)**

En este programa de C hay 5 errores que hacen que la ejecución no sea la adecuada. Encuentra dichos errores y corrige el código encima del mismo para que funcione correctamente

El programa pide al usuario introducir dos números enteros y genera un número aleatorio comprendido entre ambos. Para que el número aleatorio se genere correctamente, comprueba que el segundo número introducido sea distinto y mayor al primero. Una vez genera un número valido, pide al usuario que introduzca si cree que el número es par o impar. Lo hace pidiendo que introduzca el carácter 'p' o 'i', y valida que el valor introducido sea uno de esos caracteres. Finalmente, el programa muestra si el usuario ha acertado que era par, si ha acertado que era impar, o si ha fallado.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

int main()
{
    int num1, num2;
    printf("Introduce un numero: ");
    scanf("%d", &num1);

    //Comprueba que el segundo número es distinto al primero y mayor
    do{
        printf("\nIntroduce otro número, distinto y mayor al primero: ");
        fflush(stdin);
        scanf("%d", &num2);

    }while(num2 != num1);  //num2 <= num1

   //Generamos el numero aleatorio
    srand(time(NULL));
    int aleatorio = rand() % (num2 - num1 +1) + num1;
    printf("\nNumero generado: %d\n", aleatorio);
```

```
//Creamos un booleano para comprobar si el numero generado es par o impar
    bool par;
    if(aleatorio % 2 == 0)
    {
        par = false; //par = true
    }

    //Pedimos al usuario que introduzca el caracter 'p' o 'i' para par / impar
    char opcion;
    printf("\nIntroduce si crees que el número es par (p) o impar (i): ");
    fflush(stdin);
    scanf("%d", &opcion);  //scanf("%c", &opcion);

    //Comprobar que haya introducido 'p' o 'i'
    while (opcion != 'p' && opcion != 'i')
    {
        printf("\nCarácter invalido. Introduce si crees que es par (p) o impar (i): ");
        fflush(stdin);
        scanf("%c", &opcion);
    }


  //Comprobamos si es par o impar en base al carácter introducido por el
  //usuario (p o i)
      if(par == true && opcion == 'p')
      {
          printf("\nHas acertado! El numero %d es par", aleatorio);
      }
      else if(par == false && opcion == 'p')  //opcion == 'i'
      {
          printf("\nHas acertado! El numero %d es impar", aleatorio);
      }
      else
      {
          printf("Fallaste! El numero era %d", aleatorio);
      }
      getch();
      return 0;
  }
```

**RESUMEN ERRORES:**

**ERROR 1:** `while(num1 != num2)`

**SOLUCION:** `while(num2 == num1 && num2<num1); num<=num1`

**EXPLICACIÓN**: El enunciado y el comentario especifican que el segundo número debe ser distinto y mayor al primero.

**ERROR 2:** `int aleatorio = rand() % (num2 – num1 +1);`

**SOLUCION:** `int aleatorio = rand() % (num2 – num1 +1) + num1;`

**EXPLICACIÓN:** Si no se le suma 1, genera números entre num1 y num2 - 1

**ERROR 3:** `par = false;`

**SOLUCION:** `par = true`

**EXPLICACIÓN**: Cuando comprobamos con algún número que % 2 == 0 es para comprobar que el número sea par, por lo que el booleano par debe ser false y no true.

**ERROR 4:** `scanf("%d", &opcion);`

**SOLUCIÓN:** `scanf("%c", &opcion);`

**EXPLICACIÓN:** Se está escaneando un char, por lo que debe usarse %c

**ERROR 5:** `else if(par == false && opcion == 'p')`

**SOLUCIÓN:** `else if(par == false && opcion == 'i')`

**EXPLICACIÓN**: El if comprueba que el número sea impar, por lo que la opción debe ser ‘i’ y no ‘p’.

**EJERCICIO 2: (1 punto por pregunta, había 3 donde salían 2 aleatorias)**

**a. Define qué es un array y para qué sirve el índice de este. Pon un ejemplo.**

**Propuesta de respuesta:**

*Un array es una variable que permite almacenar varios valores del mismo tipo de dato. Los valores se pueden acceder mediante un índice que su primera posición es 0 y llega hasta donde haya definido el usuario.*

*Por ejemplo:*

*int v[10]: Es una variable que permite almacenar 10 valores del tipo entero.*

*El primer valor se situará en la posición v[0]; mientras que el último en la posición v[9].*

**b. Convierte este código con un while.**

```
int acum=0;
for(int i=0; i< =15; i++)
{
    if(i%2==0) acum=acum+4;
    if(acum> 25) break;
}
```

**Propuesta 1 de solución:**

```
int i=0, acum=0;
while(i<=15 && acum<=25){
  if(i%2==0) acum=acum+4;
  i++;
}
```

**Propuesta 2 de solución:**

```
int acum=0, i=0;
while(int i< =15){
    if(i%2==0) acum=acum+4;
    if(acum> 25) break;
    i++;
}
```

c. **Define qué es un array de cadenas y para qué sirve el índice de este. Pon un ejemplo.**

**Propuesta de respuesta:**

*Un array de cadenas nos permite almacenar en forma de variable de tipo char y en estructura matriz (array de arrays) una serie de caracteres para formar texto o palabras.*

*Estos arrays se definen por ejemplo del tipo:*

*char vArticulo[100][20]; El cual 100 sería la cantidad de productos para almacenar (jersey, pantalón, camiseta, etc) y el valor 20, sería el tamaño máximo de cada una de las palabras. No se podría almacenar más de 100 artículos ni que un artículo tuviera un tamaño en caracteres de más de 19 espacios ya que el 20º espacio es para guardar el \0.*

**EJERCICIO 3: (5,5 puntos) (cada acierto → 0,4)**

Queremos hacer un programa en C que pueda almacenar en un array de cadenas de 20 carácteres y total máximo del nombre de 100 trabajadores.

El programa preguntará al usuario cuantos trabajadores quiere introducir. Este valor debe estar validado dentro de lo permitido en el array.

Una vez sabemos cuántos trabajadores se van a introducir, se va a pedir al usuario nombre a nombre.

Posteriormente, se deben mostrar los nombres por pantalla.

```
#include <stdio.h>
#include <string.h>

#define MAXTRABAJADORES 100
#define MAXCADENA 20

int main(){

    char vTrabajadores[MAXTRABAJADORES][MAXCADENA];
    int numE = 0;


    printf("Introduce cuantos nombres de trabajadores quieres introducir:");
    fflush(stdin);
    scanf("%d",&numE);
```

```
//Comprobamos que esté dentro del rango permitido
    While(numE<1 || numE>MAXTRABAJADORES)
    {
        printf("\nNumero incorrecto.");
        printf("\nIntroduce un numero entre 1 y %d:", MAXTRABAJADORES o 100);
        fflush(stdin);
        scanf("%d",&numE);
    }
    //Pedimos al usuario los nombres

    for(int i=0; i<numE; i++)
    {
        printf("Introduce el nombre %d:",i+1);
        fflush(stdin);
        fgets(vTrabajadores[i], MAXCADENA,stdin);
    }
    //Mostramos el listado de trabajadores
    printf("\nLISTADO DE TRABAJADORES");

    for(int i=0; i<numE; i++)
    {
        printf("\nTrabajador %d: %s",i+1, vTrabajadores[i]);
    }
    printf("\nTotal trabajadores: %d",numE);


    system("pause");
    return 0;
}
```

## *UF 2: PROGRAMACIÓN MODULAR*

### EJERCICIO 1: (5,5 puntos)

En este programa de C hay 3 errores que hacen que la ejecución no sea la adecuada. Encuentra dichos errores y corrige el código encima del mismo para que funcione correctamente

El programa calcula la conversión de una cantidad de euros a dólares y libras, utilizando para ello varios procedimientos y funciones. La cantidad de euros es un número positivo introducido por el usuario.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define DOLARES 1.04
#define LIBRAS 0.86
#define MAXCADENA 20

float pideNumPositivo();
float calculaMoneda(float euros, char tipo[]);
void mostrarConversiones(float euros, float dolares, float libras);

int main()
{
    float euros;
    //Pedimos la cantidad de euros
    printf("\nIntroduce los euros a convertir:");
    euros = pideNumPositivo();

    //Hacemos el cálculo de dólares y de libras
    float dolares = calculaMoneda(euros, "dolar");
    float libras = calculaMoneda(euros, "libra");

    //Mostramos las conversiones
    float total = mostrarConversiones(euros, dolares, libras);

    mostrarConversiones(euros, dolares, libras);

    getch();
    return 0;
}
```

```
//Implementación procedimiento un número positivo

float pideNumPositivo()
{
    float num;
    printf("\nIntroduce un numero positivo (puede tener decimales): ");
    fflush(stdin);
    scanf("%d", &num);  // scanf("%f", &num);
    while (num < 0)
    {
        printf("\nNumero incorrecto. Introduce un numero positivo (puede
       tener decimales): ");
        fflush(stdin);
        scanf("%f", &num);
    }
    return num;
}

//Implementación procedimiento calculaMoneda

float calculaMoneda(float euros, char tipo) char tipo[]
{
    float conversion = 0;
    if(strcmpi(tipo, "dolar") == 0)
    {
        conversion = euros * DOLARES;
    }
    else if(strcmpi(tipo, "libra") == 0)
    {
        conversion = euros * LIBRAS;
    }
    return conversion;
}



//Implementación procedimiento mostrarConversiones

void mostrarConversiones(float euros, float dolares, float libras)
{
    printf("\n***** RESULTADO DE LAS CONVERSIONES *****\n");
    printf("\nEuros: %.2f EUR", euros);
    printf("\nDolares: %.2f $", dolares);
    printf("\nLibras: %.2f LIB", libras);
}
```

**ERRORES:**

**ERROR 1:** `float total = mostrarConversiones`

**SOLUCIÓN:** `mostrarConversiones(euros, dolares, libras);`

**EXPLICACIÓN**: mostrarConversiones es un procedimiento, es decir, no devuelve nada porque es de tipo void. Por lo tanto, solo tiene que llamarse la función y no usar el float total.
(2 puntos)

**ERROR 2:** `scanf("%d", &num);`

**SOLUCIÓN**: `scanf("%f", &num);`

**EXPLICACIÓN**: La variable num es de tipo float, por lo que hay que usar %f.

(1,5 punto)

**ERROR 3:** `float calculaMoneda(float euros, char tipo)`

**SOLUCIÓN:** `float calculaMoneda(float euros, char tipo[])`

**EXPLICACIÓN:** La variable tipo que se pasa como parámetro de la función es una cadena y no un char (puede verse en el prototipo y que en los if se comprueba con strcmpi).
(2 puntos)

**EJERCICIO 2: (4,5 puntos) (cada acierto → 0,4)**

Hacer un programa en C que, con procedimientos y paso de parámetros por valor y referencia, inicialice los 100 elementos de un array de forma aleatoria con números entre 0 y 9. Posteriormente, debe mostrarlos por pantalla.

```
#include <stdio.h>
#include <time.h>

void inicializaArray(int vNum[] o [], int *);
void pintaArray(int vNum[] o [], int );


int main(){

    int vNum[100];
    int numE=0;
    srand(time(NULL));
    inicializaArray(vNum,&numE);
    pintaArray(vNum, numE);

    getch();
    return 0;
}
```

```
void inicializaArray(int vNum[], int *numE)
{
    for(int i=0; i<100; i++)
    {
        vNum[i]= rand()%10;
        (*numE)++;
    }
}
void pintaArray(int vNum[], int numE)
{
    for(int i=0; i<numE; i++)
    {
        printf("v[%d]: %d\n",i+1,vNum[i]);
    }

}
```

## *UF 3: GESTIÓN DE FICHEROS*

**EJERCICIO 1: (1 punto por pregunta, salían 3 de 5 preguntas de forma aleatoria)**

Define para qué sirven estas funciones usadas para la gestión y control de ficheros, pon un ejemplo cuando las usarías. El ejemplo debe ser claro.

**FTELL:**

*Se utiliza para obtener la posición actual del puntero en un archivo de texto que hayamos abierto. Esto significa que se puede usar para saber en qué posición del archivo se encuentra algún carácter antes de realizar una operación de lectura o escritura.*

*Ejemplo:*

*Si nosotros sabemos hemos encontrado un carácter en concreto y gracias a ftell hemos sabido en que posición está, con la función FSEEK, podemos posicionarlos directamente allí.*

*Ftell necesita del fichero y retorna un int.*

*int pos = fell(f) donde f es un FILE.*

**REWIND:**

Rewind es una función usada en ficheros de texto para poder posicionarte al inicio del fichero. Es una función que no requiere de ningún parámetro.

Ejemplo:

En caso que nosotros hayamos ido leyendo o escribiendo en un fichero de texto y en algún momento determinado queremos ir al inicio de éste, haríamos lo siguiente: rewind().

**ATOI:**

*ATOI es una función usada en C que dada una cadena de texto (chars) convierte dicha cadena en un número entero. La cadena debe contener íntegramente números para que la conversión pueda ser correcta.*

*Ejemplo:*

*Si nosotros leemos una línea de un fichero de texto, por ejemplo con un gets o fgets, estas funciones devuelven en un array de chars (cadena) el contenido de dicha línea. Si esta línea fuera de valores numéricos, serían valores de código ASCII. No serían valores numéricos con los que pudiéramos hacer operaciones aritméticas. Por ello, debemos convertir estos números (chars) en valores int.*

```
char línea[10];
gets(linea);
int num = atoi(línea);
```

**FEOF / EOF:**

*La función FEOF(FILE) se utiliza para saber si se ha alcanzado el final del archivo (EOF) al leer un archivo especificado (FILE). Devuelve un valor distinto de cero (verdadero) si se ha llegado al final del archivo, y cero (falso) si aún quedan datos por leer.*

*EOF es una constante que indica el fin de un archivo. Se devuelve cuando un programa intenta leer más allá del final del archivo.*

*En resumen, EOF es una constante que indica el fin de un archivo y FEOF(f) es una función que te indica si has llegado al final de un archivo específico, mientras lees desde él.*

*Ejemplo:*

*Con la instrucción*

```
if( FEOF(f) == 0) printf("Sigues leyendo)
else printf("Has llegado final de fichero");
```

**FGETC / GETS**

*Fgetc: En ficheros de texto, sirve para leer un carácter de un fichero.*

*Gets En ficheros de texto, sirve para leer una línea hasta \n. Lo almacena en una variable de tipo array de chars (cadena).*

*No tiene control de overflow y no guarda el '\0', como haría la función FGETS.*

*Ejemplo de uso:*

```
FGETC:    char car = fgetc(f);
GETS:     char línea[10];
          gets(línea);
```

**EJERCICIO 2: (7 puntos) (cada acierto → 0,35 puntos)**

Hacer un programa en C para que lea el fichero de texto donde hay nombre de producto y su precio en líneas separadas.

El programa debe leer el fichero (solo leer) y mostrar por pantalla el nombre y el precio incrementado un 10%. Finalmente, debe mostrar cuantos productos se han leído.

```
#include <stdio.h>
#include <stdlib.h>

#define MAXCADENA 20
void eliminaBarraN(char []);

int main(){

    FILE *f;
    char producto[MAXCADENA];
    char precioLectura[10];
    float precio;
    int cont =0;


    f=fopen("productos.txt",”r”);


    if(f==NULL) printf("\nNo se ha podido abrir correctamente el fichero.");
    else{
        printf("\nLISTADO DE PRODUCTOS CON SUS NUEVOS PRECIOS");
        while(feof(f)==0) //Leemos hasta final de fichero
        {
            fgets(producto,MAXCADENA,f); //Leemos el nombre producto
            eliminaBarraN(producto);
            fgets(precioLectura,10,f); //Leemos el precio
            precio = atof(precioLectura); //Convertimos a float
            printf("\n%d.%s\t%.2f\t%.2f",cont+1, producto,precio,precio*1.1);
            cont++; //Actualizamos total productos
        }
        printf("\nTotal de productos listados: %d\n",cont);
    }
    system("pause");
    return 0;
}
```