**METODOLOGÍA DE LA PROGRAMACIÓN**

Grado en Ingeniería Informática
Primer curso. Segundo cuatrimestre
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

**Práctica 3: Recursividad, Ficheros y Argumentos en Línea de Órdenes.**

NOMBRE Y APELLIDOS: **VÍCTOR MONSERRAT VILLATORO.**

GRUPO: **GM8.**

**EJERCICIO 5:**

1.- ENUNCIADO.

Codifica una función recursiva que permita calcular el valor de usando la serie de Leibniz:

$$\pi = (4 – 4/3 + 4/5 – 4/7 + 4/9 – 4/11 + 4/13 \ldots)$$

El programa pedirá al usuario que introduzca el número “n” de términos a usar en la aproximación.

2.- DATOS DE ENTRADA.

- Nombre: **n.** Significado: **número de términos a usar en la aproximación de pi.** Tipo de dato: **entero (int).** Restricciones: **debe ser positivo.**

Función: **calculaValorPi (n).** Significado: **función recursiva que calcula el valor de pi aproximado a través de la serie de Leibniz.** Tipo de función: **real (double).** Restricciones: **ninguna.**

Parámetros formales:

- Nombre: **n.** Significado: **número de términos a usar en la aproximación de pi.** Significado: **número de términos a usar en la aproximación de pi.** Tipo de dato: **entero (int).** Restricciones: **debe ser positivo.**

3.- DATOS DE SALIDA.

- Nombre: **aproximación.** Significado: **aproximación del número pi.** Tipo de dato: **entero (int).** Restricciones: **ninguna.**

  **La función devuelve a la función principal por valor la aproximación de pi, tras calcularla de forma recursiva siendo lo devuelto 1.0 en el caso base, cuando n es igual a 0. La aproximación se mostrará por pantalla.**

5.- DESCRIPCIÓN DEL ALGORITMO.

**El algoritmo de la función consiste en llamarse a sí misma de forma recursiva hasta llegar al caso base (n=0) e ir sumándolo a lo ya calculado antes, calculando así la serie**

**de Leibnez para n términos. Una vez hecho esto, se multiplica el resultado final por cuatro y obtenemos una aproximación del número pi.**

**Ejemplo:**

**n=4**

**calculaValorPi (n) = pow(-1, n)/(2.0*n+1.0) + calculaValorPi (n-1)**
**calculaValorPi (n-1) = pow(-1, n-1)/(2.0*(n-1)+1.0) + calculaValorPi (n-2)**
**calculaValorPi (n-2) = pow(-1, n-2)/(2.0*(n-2)+1.0) + calculaValorPi (n-3)**
**calculaValorPi (n-3) = pow(-1, n-3)/(2.0*(n-3)+1.0) + calculaValorPi (n-4)**

**calculaValorPi (n-4) = 1 (caso base).**

**calculaValorPi (n-3) = pow(-1, n-3)/(2.0*(n-3)+1.0) + calculaValorPi (n-4) = 2.666667/4.**
**calculaValorPi (n-2) = pow(-1, n-2)/(2.0*(n-2)+1.0) + calculaValorPi (n-3) = 3.466667/4.**
**calculaValorPi (n-1) = pow(-1, n-1)/(2.0*(n-1)+1.0) + calculaValorPi (n-2) = 2.895238/4.**
**calculaValorPi (n) = pow(-1, n)/(2.0*n+1.0) + calculaValorPi (n-1) = 3.339683/4.**

**4*calculaValorPi (n) = 3.339683.**

6.- PSEUDOCÓDIGO DEL ALGORITMO.

```
Función calculaValorPi (n)
Inicio
        Si n=0
                entonces
                        Devolver 1
        Fin_si
        Sino
                entonces
                        Devuelve (-1^n/(2.0*n+1.0) + Invocar calculaValorPi (n-1))
        Fin_sino
Fin_función

Inicio

        Escribir "Introduzca el número de términos a usar en la aproximación: "
        Leer (n)
        Escribir "El valor de Pi aproximado a (n) términos es: (4*calculaValorPi (n))”
Fin
```

7.- CÓDIGO EN C.

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

double calculaValorPi (n)
{
        if (n==0)
        {
                return (1.0);
```

```
        }
        else
        {
                return (pow(-1, n)/(2.0*n+1.0) + calculaValorPi (n-1));
        }
}

int main ()
{
        int n;

        printf("\nIntroduzca el número de términos a usar en la aproximación: ");
        scanf("%i", &n);
        printf("\nEl valor de Pi aproximado a %i términos es: %lf.\n\n", n,
4*calculaValorPi (n));
        return (0);
}
```

**FUNCIÓN CONTAR NÚMERO DE REGISTROS DEL EJERCICIO 8:**

1.- ENUNCIADO.

Construye un programa que gestione mediante ficheros de texto el stock de libros de una librería. Para cada libro se almacenarán tres línea en un fichero de texto (stock); en la primera línea el título, en la segunda línea el autor, y en la tercera línea el precio y las unidades disponibles del libro. El programa contará con un menú que permitirá realizar las siguientes operaciones:

- Comprobar la existencia de un determinado libro buscando por su título.
- Introducir un nuevo libro en el stock.
- Contar el número de libros en el stock.
- Listar los libros almacenados en el stock almacenándolos previamente en un vector dinámico.
- Vender un libro buscándolo por su título.
- Borrar aquellos registros con 0 unidades disponibles.
- Salir.

2.- DATOS DE ENTRADA.

Parámetros locales:

- Nombre: **libreria.** Significado: **puntero que apunta al fichero.** Tipo de dato: **fichero (FILE *).** Restricciones: **ninguna.**
- Nombre: **libreria.** Significado: **puntero que apunta al fichero.** Tipo de dato: **fichero (FILE*).** Restricciones: **ninguna.**

3.- DATOS DE SALIDA.

- Nombre: **n.** Significado: **número de libros totales que hay en stock en la librería.** Tipo de dato: **entero (int).** Restricciones: **ninguna.**

  **La función devuelve el parámetro n a la función principal por valor, en la cual se mostrará por pantalla.**

4.- DATOS AUXILIARES.

Parámetros locales:

- Nombre: **salir.** Significado: **variable para salir del bucle cuando termina de leer el fichero entero.** Tipo de dato: **entero (int).** Restricciones: **ninguna.**
- Nombre: **contador.** Significado: **variable para comprobar que el número de datos leídos es correcto.** Tipo de dato: **entero (int).** Restricciones: **ninguna.**
- Nombre: **stock.** Significado: **variable para ir sumando el stock de cada uno de los libros.** Tipo de dato: **entero (int).** Restricciones: **debe ser positivo.**
- Nombre: **EOF.** Significado: **constante definida en stdlib.h como -1.** Tipo de dato: **entero (int).** Restricciones: **ninguna.**
- Nombre: **titulo.** Significado: **variable para almacenar los titulos de los libros.** Tipo de dato: **cadena de caracteres (char*).** Restricciones: **ninguna.**
- Nombre: **autor.** Significado: **variable para almacenar los autores de los libros.** Tipo de dato: **cadena de caracteres (char*).** Restricciones: **ninguna.**
- Nombre: **precio.** Significado: **variable para almacenar los precios de los libros.** Tipo de dato: **real (float).** Restricciones: **debe ser positivo.**

5.- DESCRIPCIÓN DEL ALGORITMO.

**La función va leyendo del fichero todos los stocks de los libros que hay almacenados a través de un bucle y los va almacenando en una variable que devuelve al programa principal y este la imprime por pantalla.**

**Ejemplo:**

**Fichero:**

**Matemáticas**
**Gauss**
**23.000000 2**
**Evolución**
**Darwin**
**21.000000 4**

**Salida:**

**2+4 = 6.**

6.- PSEUDOCÓDIGO DEL ALGORITMO.

```
Función contarLibros ()
Inicio
    Abrir “Librería.txt” en modo lectura
    Mientras salir = 0
        entonces
            contador ← Leer (titulo, autor, precio, stock) de “Librería.txt”
            Si contador = EOF
                entonces
                    salir ← 1
            Fin_si
```

```
                Sino
                        entonces
                                n ← n+stock
                Fin_sino
        Fin_mientras
        Cerrar “Librería.txt”
        Devolver (n)
Fin_función
```

7.- CÓDIGO EN C.

```
        int contarLibros ()
        {
                FILE* libreria;
                int salir=0, contador, stock, n=0;
                char titulo [DIM_CHAR], autor [DIM_CHAR];
                float precio;

                libreria = fopen("Librería.txt", "r");
                while (salir == 0)
                {
                        contador = fscanf (libreria, "%s%s%f %i", titulo, autor, &precio,
        &stock);
                        if (contador == EOF)
                        {
                                salir=1;
                        }
                        else
                        {
                                n = n+stock;
                        }
                }
                fclose (libreria);
                return (n);
        }
```