

# **DESARROLLO DE APLICACIONES MULTIPLATAFORMA PROGRAMACIÓN**

## TEMA 1

# **PROGRAMACIÓN ESTRUCTURADA**

(Lenguaje de Programación C)

# CONTENIDOS

- 1.1. Programación Estructurada.
- 1.2. Lenguaje de Programación C.
- 1.3. Compilador de C.
- 1.4. Primer programa en C.
- 1.5. Tipos de datos.
- 1.6. Variables, constantes y asignaciones.
- 1.7. Escritura y lectura por consola.
- 1.8. Operadores aritméticos, comparación y lógicos.
- 1.9. Condicionales.
- 1.10. Bucles.
- 1.11. Funciones.

## 1.1. PROGRAMACIÓN ESTRUCTURADA.

La Programación Estructurada es un paradigma de la programación, orientada a mejorar la claridad, calidad y tiempo de desarrollo de un programa de ordenador recurriendo únicamente a funciones y tres estructuras básicas: secuencia, selección e iteración.

El Teorema del Programa Estructurado, demuestra que todo programa puede ser escrito únicamente utilizando las tres instrucciones de control:

- *Secuencia (sucesión de instrucciones)*
- *Selección (instrucción condicional).*
- *Iteración (bucle de instrucciones con condición inicial o final).*

Solamente con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, éstas pueden ser construidas mediante las tres básicas.

## 1.2. LENGUAJE DE PROGRAMACIÓN C.

El Lenguaje de Programación C fue desarrollado por Dennis Ritchie entre 1969 y 1972, estandarizado por ANSI en 1989 y ratificado por ISO en 1990.

Es un lenguaje orientado a la implementación de Sistemas Operativos (*Unix*). Es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también para crear aplicaciones.

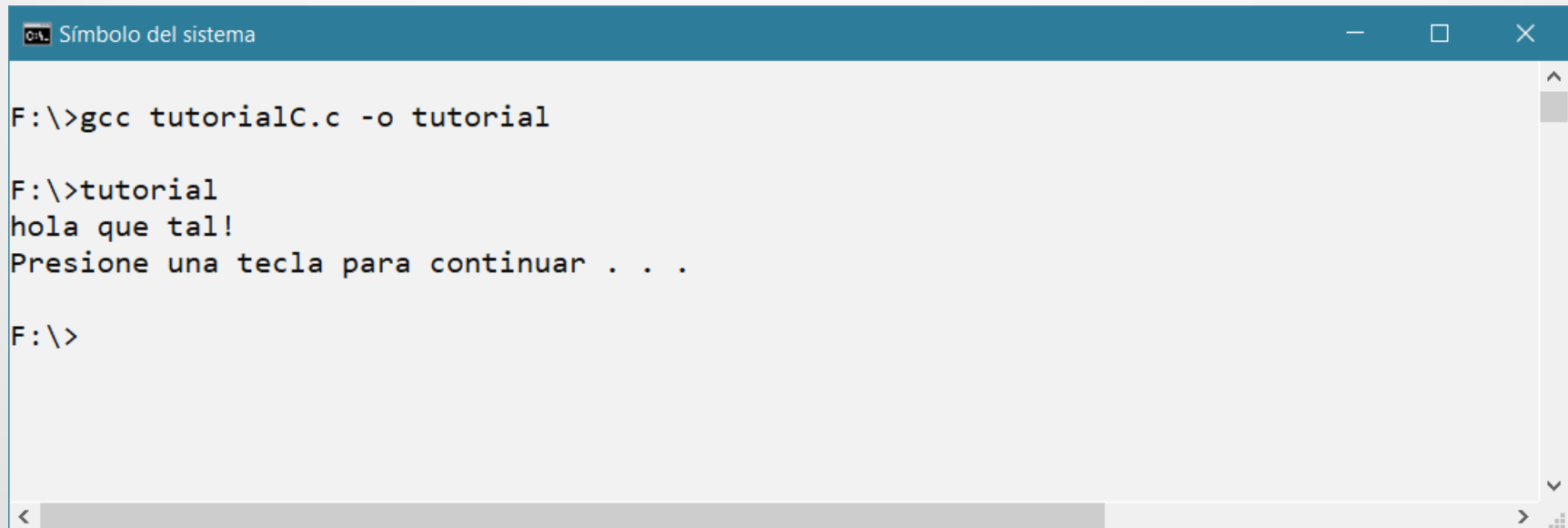
Se trata de un lenguaje de tipos de datos estáticos, débilmente tipado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones que permiten un control a muy bajo nivel (acceso a memoria, acceso a periféricos, ...).

## 1.3. COMPILADOR DE C.

El Compilador de C es una herramienta que nos permite convertir nuestro programa en un ejecutable entendible por el ordenador.

Existe una gran variedad de compiladores: *Turbo C++*, *DJGPP*, *Borland C++*, *DevC++*,... Nosotros utilizaremos MinGW (*Minimalist GNU for Windows*).

Ejemplo:



```
0% Símbolo del sistema

F:\>gcc tutorialC.c -o tutorial

F:\>tutorial
hola que tal!
Presione una tecla para continuar . . .

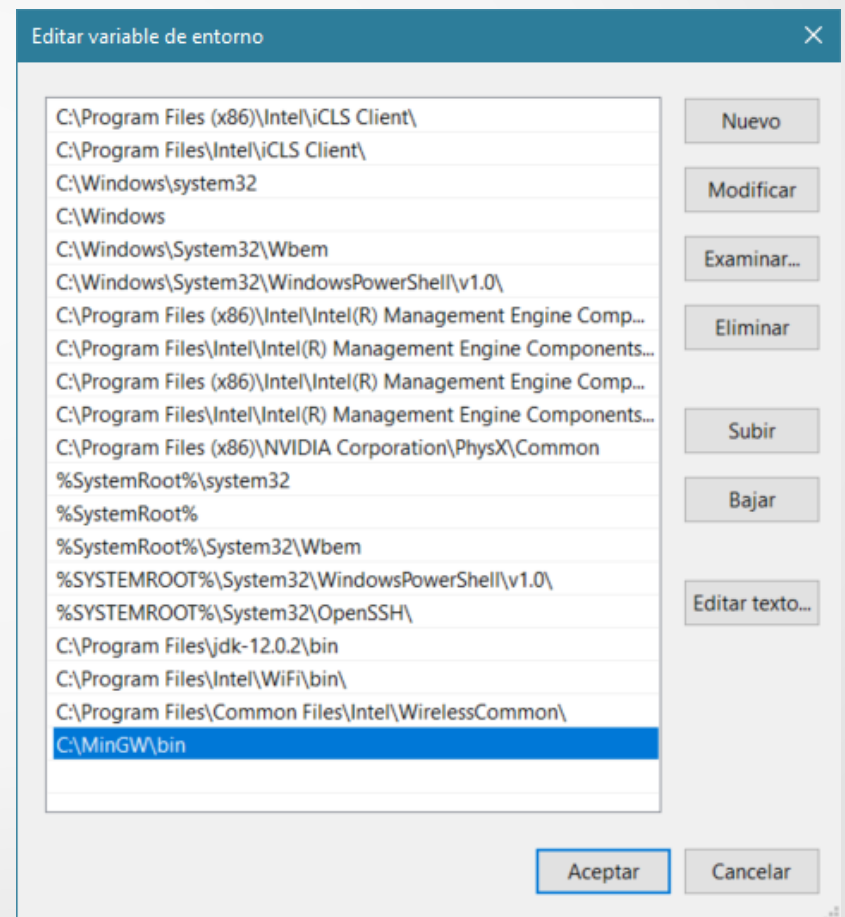
F:\>
```

## 1.3. COMPILADOR DE C.

Para instalar MinGW, descomprimir el fichero ZIP suministrado por el profesor en la raíz del sistema de archivos Windows unidad C:, de tal forma que aparezca una carpeta llamada C:\MinGW. Dentro de esta carpeta está todo lo necesario para compilar y ejecutar nuestros programas en Lenguaje C.

Para configurar MinGW y poder utilizar la herramienta de compilación “gcc” desde la Consola de Windows, hay que añadir la carpeta “C:\MinGW\bin” al PATH del sistema.

*Inicio→Sistema de Windows→Panel de control  
→Sistema y seguridad→Sistema  
→Configuración avanzada del sistema  
→Variables de entorno  
→Variables del sistema→Path*



## 1.4. PRIMER PROGRAMA EN C.

```
#include <stdio.h>           // Uso de librería
#include <stdlib.h>          // Uso de librería

int main() {                 // Función principal
    // Las llaves indican un bloque de código
    // Los punto y coma indican fin de sentencia

    printf("hola que tal!\n"); // Mostrar por pantalla

    system("pause");          // Pausa la consola de Windows
    return 0;
}
```

## 1.5. TIPOS DE DATOS.

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

int main() {

    short    num1 = 0;           // Tipo entero (2 bytes -128..127)
    int       num2 = 1;          // Tipo entero (2 bytes -32768..32767)
    long      num3 = 2;          // Tipo entero (4 bytes -2147483648..2147483647)
    float      num4 = 1.2;        // Tipo decimal (4 bytes)
    double     num5 = 2.3;        // Tipo decimal (8 bytes)
    char       letra = 'a';       // Tipo carácter (1 byte 0..255)

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



## 1.6. VARIABLES, CONSTANTES Y ASIGNACIONES.

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

int main() {

    int x;                // Variable
    const int y = 3.1415; // Constante

    x=0;                  // Asignación

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

## 1.7. ESCRITURA Y LECTURA EN CONSOLA.

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

int main() {

    char letra = 'a';
    int num1 = 1;
    float num2 = 2.15;

    printf("La caracter es: %c\n",letra);
    printf("Los numeros son: %i y %f\n",num1,num2);

    // Modificadores
    // %i para enteros    %f para decimales
    // %c para caracteres %s para cadenas
    // Flags
    // - justifica a la izquierda + muestra el signo
    // 0 rellena con ceros x.y precisión de decimales

    printf("El numero decimal es: %+02.1f\n",num2);

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

## 1.7. ESCRITURA Y LECTURA EN CONSOLA.

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

int main() {

    char letra;
    int num;

    printf("Introduce un numero: ");
    scanf("%i",&num);
    printf("El numero es: %i\n",num);

    printf("Introduce una letra: ");
    scanf(" %c",&letra);           // !!
    printf("La letra es: %c\n",letra);

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

## 1.8. OPERADORES (ARITMÉTICOS, COMPARACIÓN Y LÓGICOS).

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

int main() {

    // Suma + Resta - Multiplicación * División /
    // Incremento ++ Decremento -- Resto %

    int x,y;

    x = 1;
    y = (x + 1) - 2;    // Uso de paréntesis (prioridad)

    ++x;               // Similar a x = x + 1
    x++;               // !!

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

## 1.8. OPERADORES (ARITMÉTICOS, COMPARACIÓN Y LÓGICOS).

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

int main() {

    // == (igual que)    != (distinto a)
    // < (menor que)    > (mayor que)
    // <= (menor o igual que)    >= (mayor o igual que)

    int x = (1==1);
    int y = (1!=1);

    printf("1 es igual a 1? %i\n",x);
    printf("1 es distinto de 1? %i\n",y);

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

## 1.8. OPERADORES (ARITMÉTICOS, COMPARACIÓN Y LÓGICOS).

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

int main() {

    // && (y lógico)
    // || (o lógico)
    // ! (no)

    int x=1<2 && 2<3;

    printf("1 es menor que 2 y 2 es menor que 3? %i\n", x);
    printf("1 es menor que 2 o 1 es mayor que 2? %i\n", 1<2 || 1>2);

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

## 1.9. CONDICIONALES (IF., IF..ELSE., IF..ELSE IF..ELSE..).

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

int main() {

    int num;

    printf("Introduce un numero: ");
    scanf("%i", &num);

    if (num >= 0) {
        printf("El numero %i es positivo.\n", num);
    }

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

## 1.9. CONDICIONALES (IF., IF..ELSE., IF..ELSE IF..ELSE..).

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

int main() {

    int num;

    printf("Introduce un numero: ");
    scanf("%i", &num);

    if (num >= 0) {
        printf("El numero %i es positivo.\n", num);
    } else {
        printf("El numero %i es negativo.\n", num);
    }

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



## 1.9. CONDICIONALES (IF., IF..ELSE., IF..ELSE IF..ELSE..).

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

int main() {

    int num;

    printf("Introduce un numero: ");
    scanf("%i", &num);

    if (num>0) {
        printf("El numero %i es positivo.\n", num);
    } else if (num<0) {
        printf("El numero %i es negativo.\n", num);
    } else {
        printf("El numero es cero.\n");
    }

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

## 1.9. CONDICIONALES (SWITCH..).

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

int main() {

    int num;

    printf("Introduce un numero: ");
    scanf("%i", &num);

    switch (num) {
        case 1:
            printf("El numero es el uno.\n");
            break;
        case 2:
            printf("El numero es el dos.\n");
            break;
        default:
            printf("El numero es otro.\n");
            break;
    }

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

## 1.10. BUCLES (WHILE..).

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

int main() {

    int num=1;

    while (num<5) {
        printf("%i\n",num);    // Muestra del 1 al 4
        num++;
    }

    while (num) {
        printf("%i\n",num);    // Muestra del 5 al 9
        num++;
        if (num==10) {
            num=0;
        }
    }

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

## 1.10. BUCLES (DO..WHILE).

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

int main() {

    int num=1;

    do {
        printf("%i\n",num);           // Muestra del 1 al 4
        num++;
    } while (num<5);

    do {
        printf("%i\n",num);           // Muestra del 5 al 9
        num++;
        if (num==10) {
            num=0;
        }
    } while (num);

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

## 1.10. BUCLES (FOR..).

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

int main() {

    for (int i=1; i<=5; i++) {
        printf("%i\n",i);           // Muestra del 1 al 5
    }

    for (int i=0; i<=9; i++) {
        for (int j=0; j<=9; j++) {
            printf("%i%i\n",i,j);   // Muestra del 00 al 99
        }
    }

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

## 1.11. FUNCIONES (*SIN PARÁMETROS*).

```
/*  
    FUNCIONES SIN PARÁMETROS  
*/  
  
#include <stdio.h>  
#include <stdlib.h>  
  
void mostrar() {                                // Función sin parámetros ni retorno  
    printf("hola\n");  
    printf("adios\n");  
}  
  
int main() {  
    mostrar();  
  
    system("pause");  
    return 0;  
}
```

## 1.11. FUNCIONES (CON PARÁMETROS).

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

int suma(int x, int y) {           // Función con parámetros y retorno
    int resultado=x+y;
    return resultado;
}

int main() {

    int a,b;
    int c;

    printf("Introduce un numero: ");
    scanf("%i",&a);
    printf("Introduce otro numero: ");
    scanf("%i",&b);

    c=suma(a,b);                  // Ejemplo 1
    printf("La suma de %i y %i es %i\n",a,b,c);

    printf("La suma de %i y %i es %i\n",1,2,suma(1,2)); // Ejemplo 2

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