Universidad Nacional Autónoma de México
Facultad de Ingeniería

Microprocesadores y Microcontroladores

Tema 8

Lenguaje C

M. I. Christo Aldair Lara Tenorio

2025-1

Tabla de contenidos

Objetivo del tema

Objetivo general:
El alumno programará en lenguaje C.

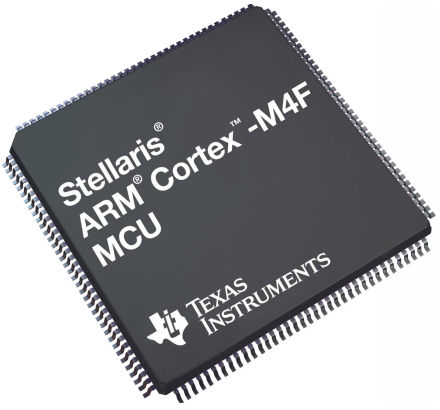
Contenido:

8.1. Introducción histórica de C.

8.2. Conceptos básicos.

8.3. Estructuras de control de flujo y tiempo.

8.4. Ejemplos de programación.



Introducción histórica del Lenguaje C

1969 – 1970 **Lenguaje B**: Desarrollado por Ken Thompson en los Laboratorios Bell como un derivado simplificado del lenguaje BCPL (Basic Combined Programming Language), diseñado para escribir sistemas operativos y aplicaciones en el primer sistema UNIX.

1972 **Creación del lenguaje C**: Desarrollado por Dennis Ritchie, está basado en B, añadiendo características como tipos de datos, punteros y mayor control sobre la memoria.

- C fue creado para desarrollar el sistema operativo UNIX (previamente escrito en ensamblador), agregándole portabilidad y un mayor mantenimiento.
- C se convirtió en el lenguaje principal para el desarrollo de sistemas operativos, controladores de dispositivos y herramientas de software.

1978 Publicación del libro "The C Programming Language", de Brian Kernighan y Dennis Ritchie (K&R C).

- Referencia definitiva para aprender y entender C.

1983 El Instituto Nacional Estadounidense de Estándares (ANSI) forma un comité para desarrollar un estándar oficial para C.

1989 Publicación del estándar **ANSI C**, conocido como **C89**.

Introducción histórica del Lenguaje C

1990	Adopción del estándar oficial para C por la Organización Internacional de Normalización (ISO) como **C90**, unificando el lenguaje y asegurando su consistencia y portabilidad en diferentes plataformas.
1999	Versión **C99**: múltiples mejoras al lenguaje. • Declaración de variables en cualquier lugar del código. • Nuevos tipos de datos (`long long int`). • Manejo de arreglos de longitud variable y mejoras en las funciones matemáticas.
2011	Versión **C11**: mejoras en la gestión de la concurrencia, alineación de memoria y mayor soporte para sistemas multinúcleo.
2018	Versión **C18**: Revisión de C11 para corregir errores y pequeñas actualizaciones para mejorar la implementación del lenguaje.

Impacto:

- Múltiples lenguajes derivados (C++, Objective-C, C#, Java, JavaScript, Go).
- Lenguaje dominante en sistemas operativos (Linux, Windows y gran parte de macOS), dispositivos embebidos y controladores de hardware.
- Aplicaciones criticas y de alto rendimiento (bases de datos, sistemas de redes y motores de juegos).

Lenguaje C

Lenguaje de propósito general, estructurado y de medio nivel que es conocido por su eficiencia, portabilidad y flexibilidad.

Características principales:

- Lenguaje de medio nivel.
  Tiene estructuras de control (alto nivel) y permite manipular directamente el hardware (bajo nivel).
- Eficiente.
  Permite un control preciso sobre la memoria y los recursos de hardware del sistema.
- Estructurado.
  Permite descomponer el código en funciones más pequeñas, manteniendo una alta legibilidad.
- Portabilidad.
  Puede ser ejecutado en múltiples dispositivos/arquitecturas (siempre y cuando se tenga el compilador adecuado).
- Acceso directo a memoria.
  Permite definir punteros para modificar/leer datos directamente en memoria.

Compilador

Programa que traduce el código fuente escrito en un lenguaje de programación de alto nivel al código máquina, de tal modo que este pueda ser ejecutado por un procesador.

- **Revisión de la sintaxis** → Descomposición del código fuente en bloques más pequeños para determinar que las palabras clave y operandos cumplen con las reglas gramaticales del lenguaje.
- **Análisis semántico** → Verificación de que el código fuente tenga sentido en el contexto del lenguaje.
- **Optimización** → Incrementar la eficiencia del código.
- **Generación de código** → Traducción del código a bajo nivel (código máquina).

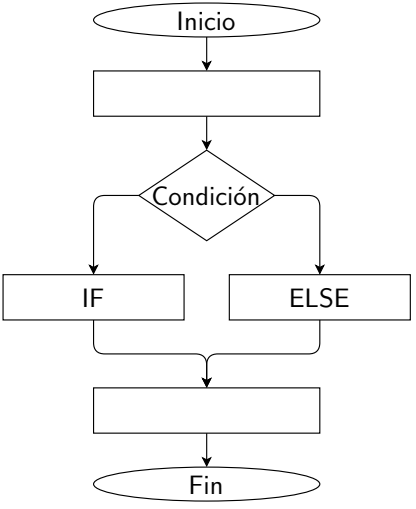
Lenguaje C	Lenguaje ensamblador	Código máquina
GPIO_PORTF_DIR_ |= 0x11;	LDR R1, [PC, #0xA0]	4928
	LDR R1, [R1]	6808
	ORR R0, R0, #0x11	F0400011
	STR R0, [R1]	6008

Estructura básica de un programa en lenguaje C para microcontroladores

1. Archivos de cabecera.
   Definición de los archivos que contienen las declaraciones de funciones, macros y tipos de datos que se van a utilizar en el código.
2. Definiciones y macros.
   Se utiliza el macro `#define` para definir constantes relacionadas con el hardware, como direcciones de memoria.
3. Funciones para la inicialización de los periféricos del microcontrolador.
   Funciones en donde se programa el proceso de inicialización y configuración de los periféricos del microcontrolador.
4. Función principal (`main`(void)).
   Función desde la cual se mandan a llamar todas las funciones de inicialización y congela el proceso en un ciclo infinito en donde se ejecutarán las instrucciones de control del sistema.

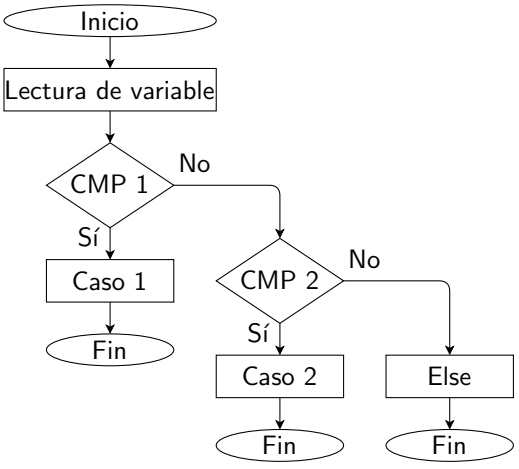
Estructuras de control: IF - ELSE

```
01 if (condición1) {
02     // Código a ejecutar si condición1 es
   verdadera
03 } else if (condición2) {
04     // Código a ejecutar si condición2 es
   verdadera
05 } else {
06     // Código a ejecutar si ninguna de las
   condiciones es verdadera
07 }
```



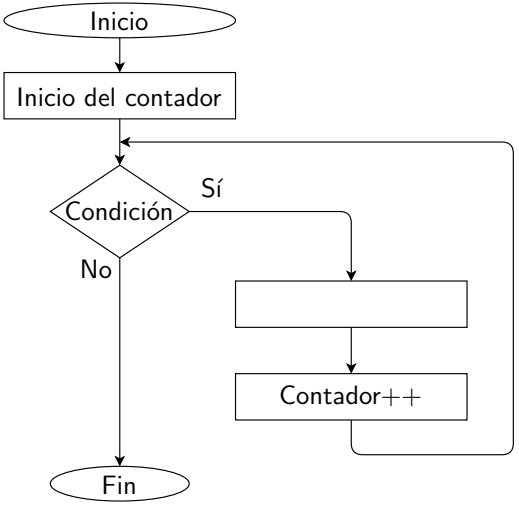
Estructuras de control: SWITCH-CASE

```
01 switch (expresión) {
02     case valor1:
03         // Código a ejecutar si expresión es igual a valor1
04         break;
05     case valor2:
06         // Código a ejecutar si expresión es igual a valor2
07         break;
08     default:
09         // Código a ejecutar si ninguno de los casos anteriores coincide
10 }
```



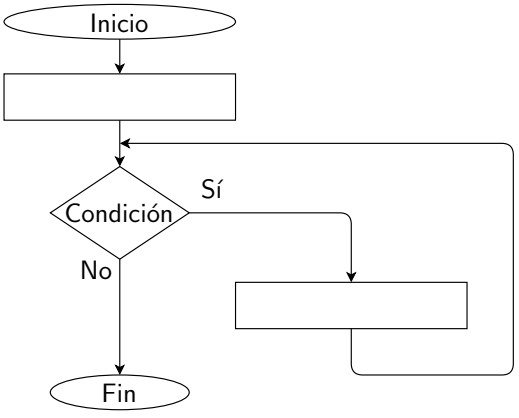
Estructuras de control: FOR

```c
for (inicialización; condición; incremento) {
    // Código a ejecutar en cada iteración
}
```

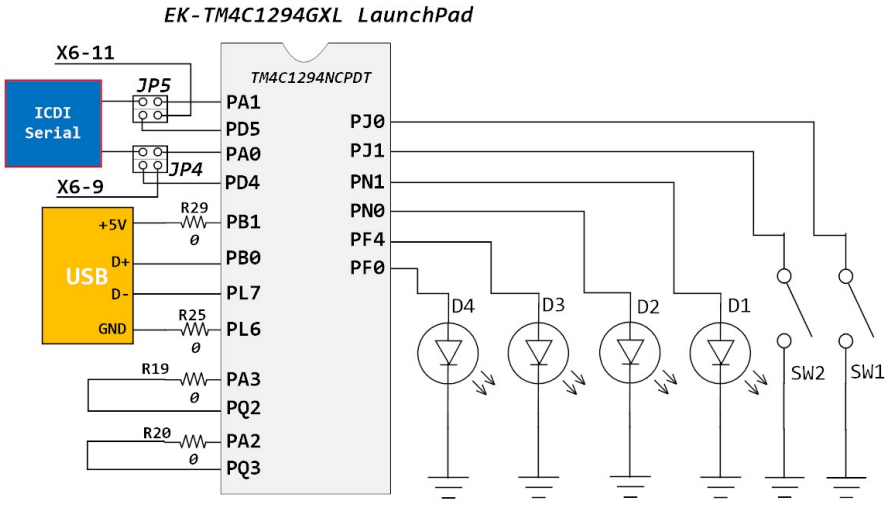


Estructuras de control: WHILE

```
01 | while (condición) {
02 |     // Código a ejecutar mientras la condición
   |  sea verdadera
03 | }
```



LED y botones de la tarjeta de desarrollo



Tarea 6 – GPIO con lenguaje C

Desarrollar los códigos en lenguaje C que realicen lo siguiente:

C01: (**Código 14**) Encender y apagar los LED de usuario de la tarjeta de desarrollo en el siguiente orden, con una frecuencia de encendido/apagado de $1Hz$, repitiendo el proceso 5 veces.

1. `LED1` $\rightarrow$ *ON*
2. `LED2` $\rightarrow$ *ON*
3. `LED3` $\rightarrow$ *ON*
4. `LED4` $\rightarrow$ *ON*
5. `LED1` $\rightarrow$ *OFF*
6. `LED2` $\rightarrow$ *OFF*
7. `LED3` $\rightarrow$ *OFF*
8. `LED4` $\rightarrow$ *OFF*

C02: Encender y apagar los LED de usuario de la tarjeta de desarrollo, dependiendo de los interruptores que se presionen.

- Si se presiona el `SW1`, conmutar el `LED1`.
- Si se presiona el `SW2`, conmutar el `LED2`.

* Considerar un retardo de rebote inicial de $10ms$ y ajustarlo al que presente el mejor comportamiento.

TivaWare

Conjunto extenso de bibliotecas de software, controladores y ejemplos de Texas Instruments.

- Diseñado para simplificar el desarrollo de aplicaciones en los microcontroladores de la familia **TIVA C Series**.
- Permite que los desarrolladores se concentren más en el diseño de la aplicación que en la configuración y control a bajo nivel del hardware.

Componentes del TivaWare:

- Driverlib → Biblioteca de controladores para interactuar con los periféricos del microcontrolador.
- TIVA Peripheral Driver Library → Conjunto de controladores que permite el acceso directo a los periféricos con funciones simples.
- USB Library → Soporte para aplicaciones que usan el controlador USB en modo dispositivos, host o OTG.
- Graphics Library (grlib) → Biblioteca que permite diseñar y manejar interfaces gráficas (GUI) en pantallas LCD.
- Examples → Amplia gama de ejemplos de código que cubren la mayoría de los periféricos y configuraciones comunes.

Ventajas:

- Facilita el mantenimiento de código e incrementa la portabilidad entre dispositivos de la familia TIVA C Series.
- Incrementa la eficiencia en el desarrollo de software.
- API intuitiva y bien documentada para facilitar su uso.

Modelos de programación:

- Modelo de acceso directo a registro (DRA)
  Enfoque de programación de bajo nivel que permite al programador interactuar directamente con los registros de hardware del microcontrolador.
  - Alta eficiencia y rendimiento.
  - Mayor complejidad en el proceso de diseño.
  - Dependiente del hardware.
- Modelo de Software
  Se basa en el uso de bibliotecas de software o drivers para proporcionar una abstracción de los registros de hardware.
  - Eficiencia y rendimiento moderados.
  - Mayor facilidad de su uso.
  - Mayor portabilidad.