
PUCP

PROGRAMACIÓN DE MICROCONTROLADORES ARM

CETAM - PUCP

Datos del docente

- Nombre : Alexander Francisco Segovia Razo
- Nro de contacto : +51 982 495 578
- Email : afsegovia@pucp.edu.pe

  francisco.segovia.razo@gmail.com

Calendario del curso

- Jueves 10 de agosto
- Martes 15 de agosto
- Jueves 17 de agosto
- Martes 22 de agosto
- Examen parcial
  - 22/08 - 29/08
- Jueves 24 de agosto
- Martes 29 de agosto
- Jueves 31 de agosto
- Martes 5 de septiembre
- Examen final
  - 05/09 - 10/09

Objetivos del curso

- Profundizar en el lenguaje de programación C para aplicarlo a los microcontroladores con arquitectura ARM
- Conocer el ciclo de vida de un proyecto de desarrollo embebido y aplicar las herramientas apropiadas para brindarle mantenimiento y documentarlo de forma apropiada
- Conocer los periféricos internos de un microcontrolador ARM, sus aplicaciones y la forma de programarlos
- Explorar las aplicaciones de los sistemas embebidos

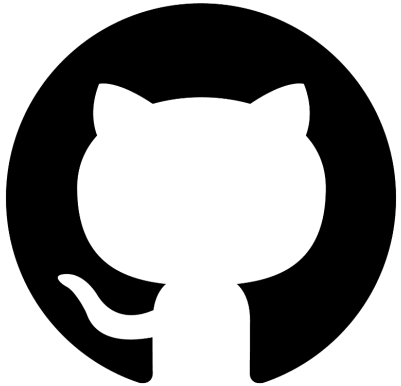
¿Que es un desarrollador embebido?

¿Que es un desarrollador embebido?

Herramientas de desarrollo : C

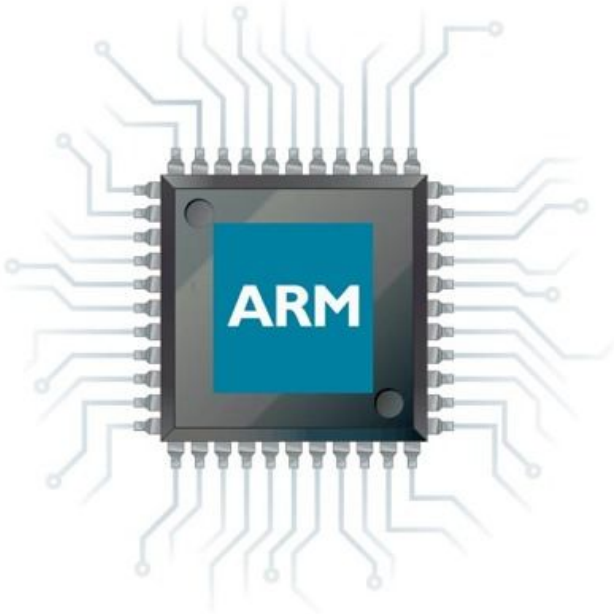
Herramientas de documentación

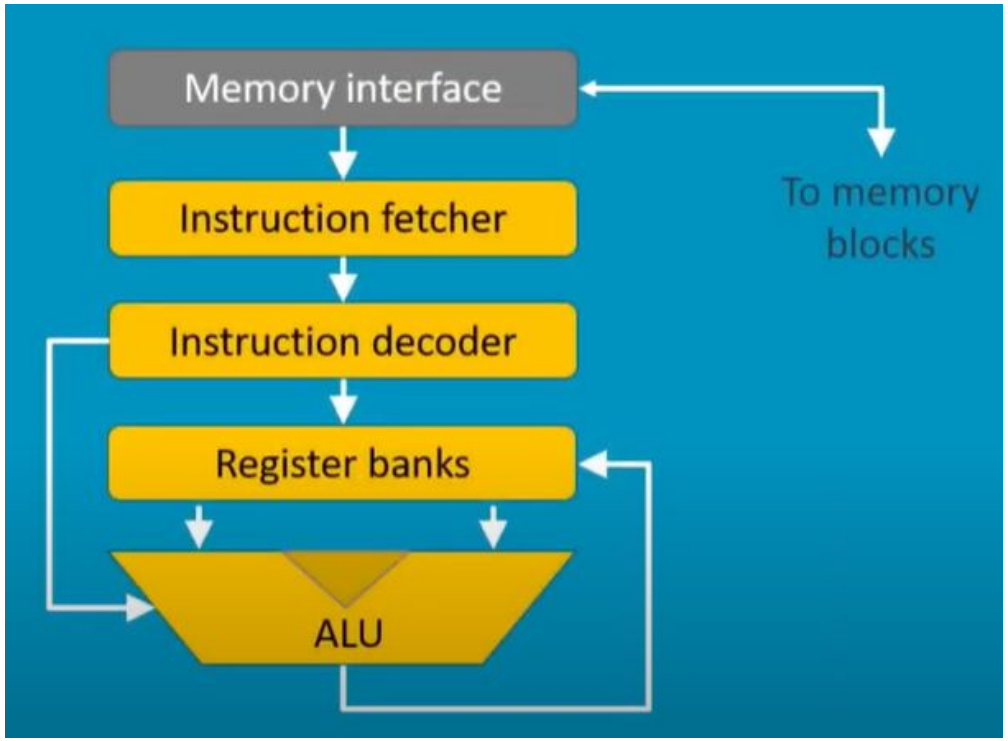
git



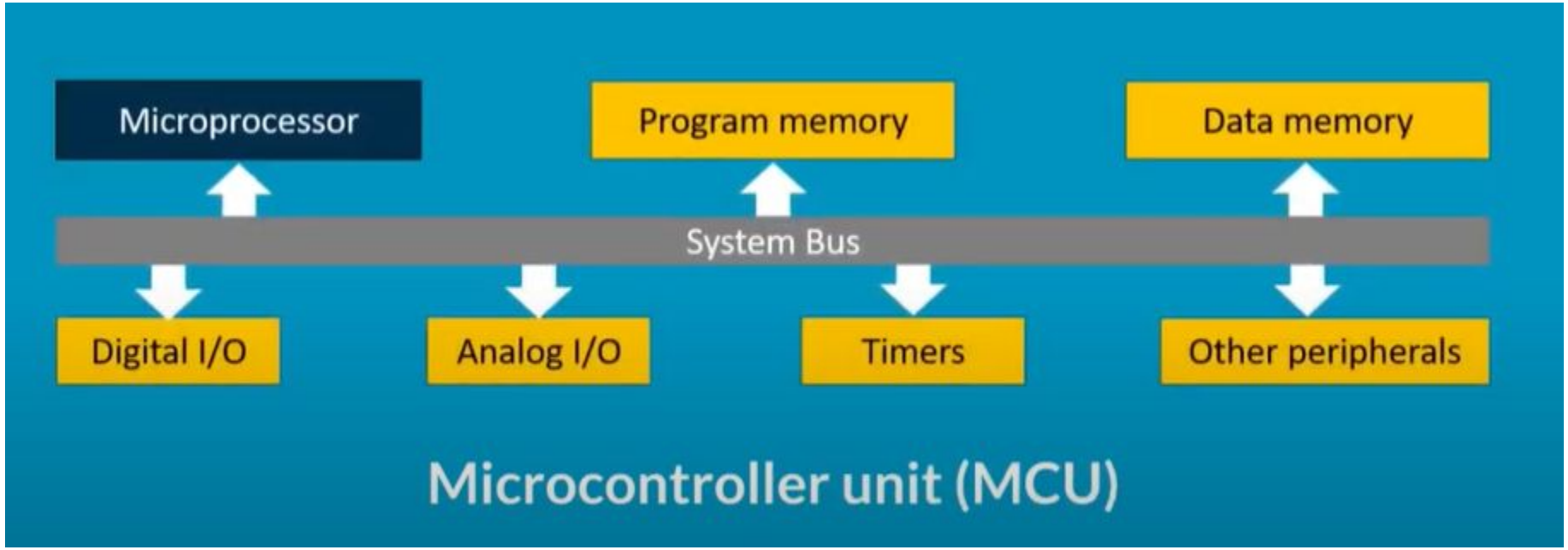
GitHub

Microcontroladores





Periféricos de un microcontrolador



Aplicaciones

¿Cómo convertirnos en desarrolladores embebidos?

SILABO DEL CURSO

- Lenguaje de programación C
- Sistema de control de versiones: GIT
- Procesadores embebidos ARM
- Periféricos programables I: Puertos E/S de propósito general
- Periféricos programables II: Módulo ADC
- Periféricos programables III: Temporizadores
- Protocolos de comunicación: Módulo USART, SPI e I2C
- Control de interrupciones
- Programación orientada a objetos y uso de librerías

Herramientas del curso

- Se utilizarán simuladores como MBED y compiladores de código abierto
- Las evaluaciones serán entregadas a través de los repositorios personales Git de cada estudiante
- La documentación del curso y las sesiones serán subidas a la plataforma CANVAS

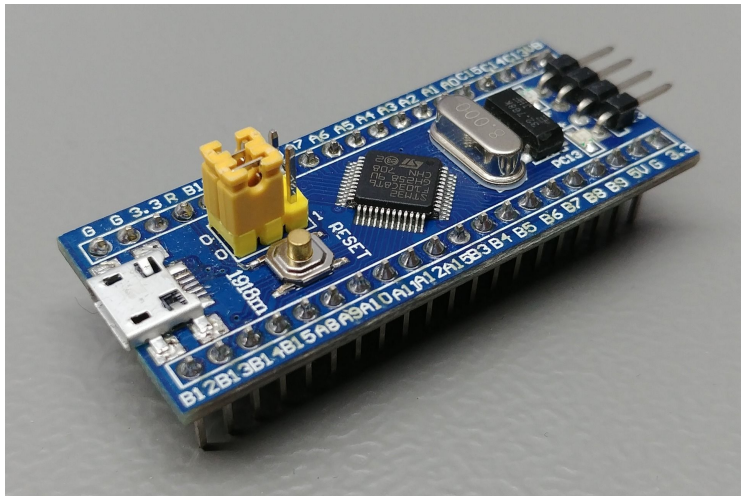
Evaluación del curso

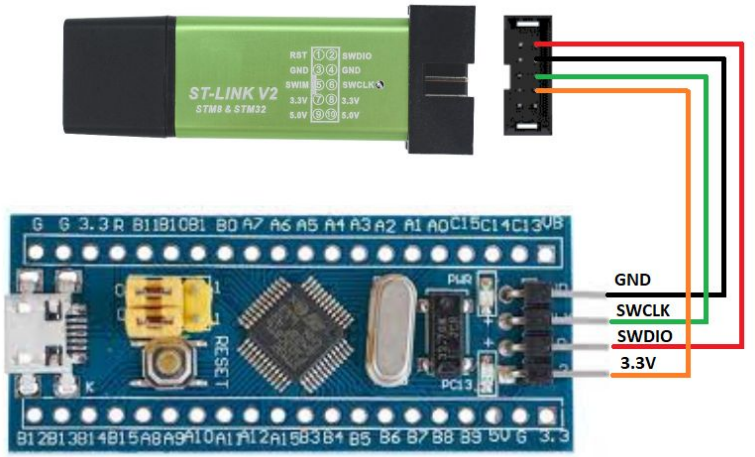
La evaluación del curso será de la siguiente forma y porcentajes:

- Examen parcial : 40%
- Examen final : 45%
- Asistencia : 15%

Tarjetas de desarrollo

Para el presente curso se utilizará la tarjeta STM32F103C8T6, el cual necesita de un módulo adicional ST-LINK para poder programarlo. Estos módulos están disponibles en el mercado nacional para su adquisición.





Tarjetas de desarrollo

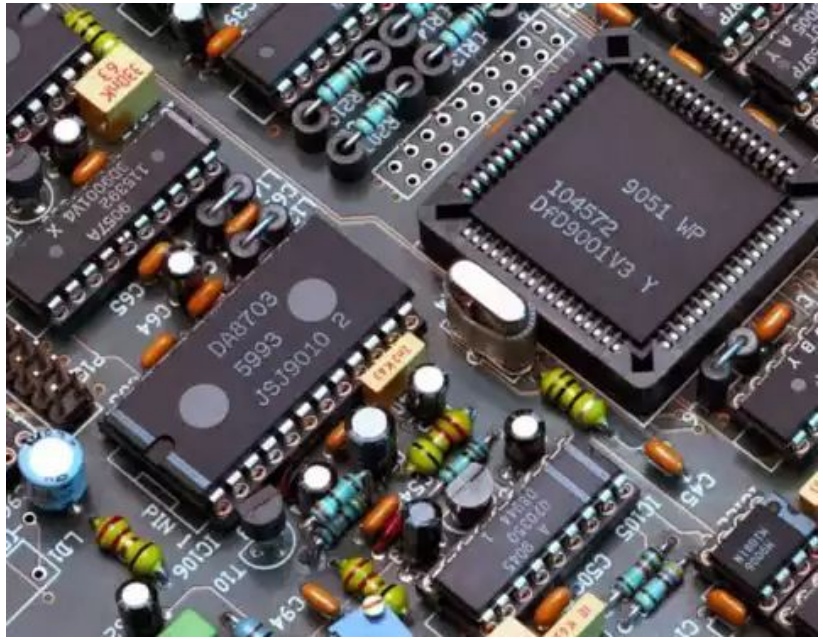
Otra opción a tener en cuenta es la tarjeta de desarrollo Raspberry Pi Pico; sin embargo esta tarjeta sólo será utilizada para aplicaciones con lenguaje de alto nivel

Oportunidades laborales

En el campo del desarrollo embebido, desarrollo web, desarrollo móvil, inteligencia artificial se tienen oportunidades remotas las cuales pueden ser encontradas en los siguientes portales. Las características se pueden observar en sus descripciones. Siempre tener en cuenta que es muy importante acumular experiencia en el campo. En el presente curso se brindarán las herramientas principales para que los participantes puedan seguir explorando de forma autónoma el campo de los sistemas embebidos.

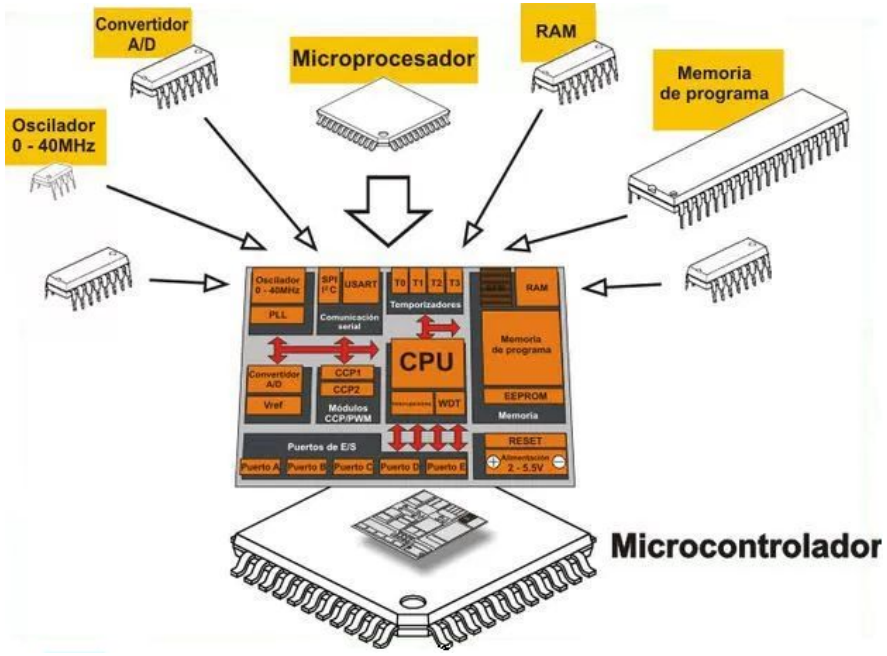
¿Qué es un microcontrolador?

Al interior de todos los dispositivos electrónicos que nos rodean se tienen como elementos de control circuitos integrados conocidos como microcontroladores, los cuales son sistemas digitales programables que interactúan con actuadores y sensores.



Microcontrolador o Microprocesador

Un microcontrolador es un concepto más general en comparación a un Microprocesador ya que se integra con diferentes periféricos que permiten su interacción con el entorno de desarrollo.



¿Qué es ARM?

ARM son las siglas de Advanced RISC Machine y representan a todos aquellos microcontroladores que utilicen dicho set de instrucciones. RISC permite una mayor eficiencia energética en el procesamiento de información. En comparación con arquitecturas x86, los ARM permiten una larga vida a las baterías de los equipos

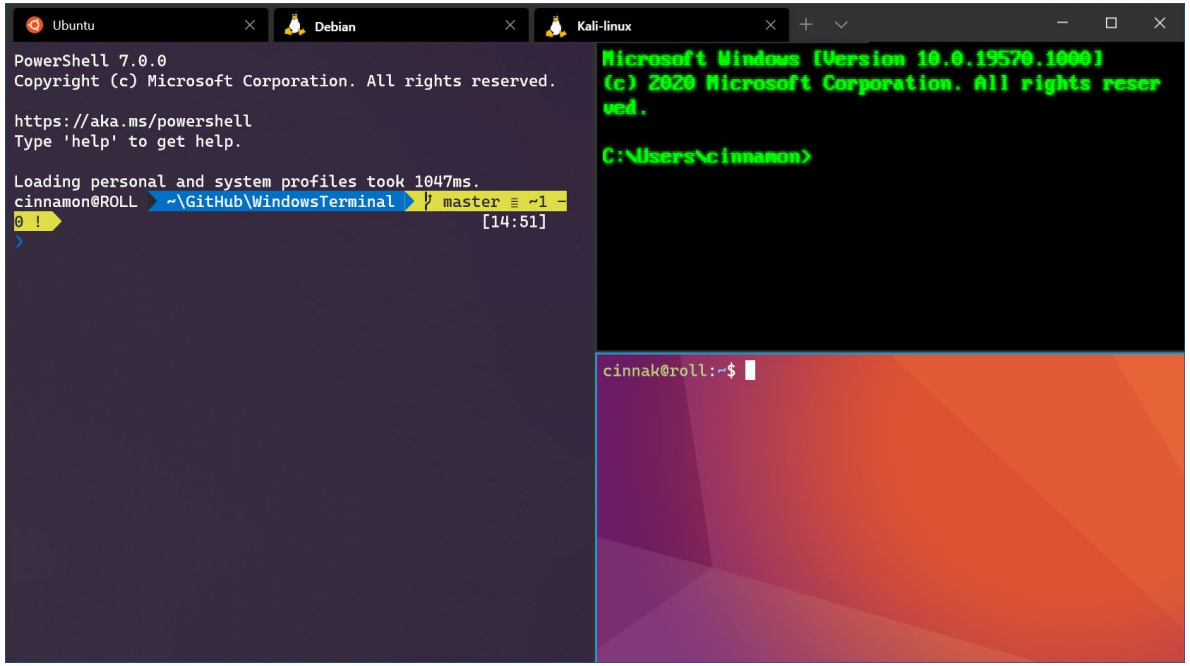
Consola de comandos

Programación de microcontroladores ARM - Sesión 1

PUCP

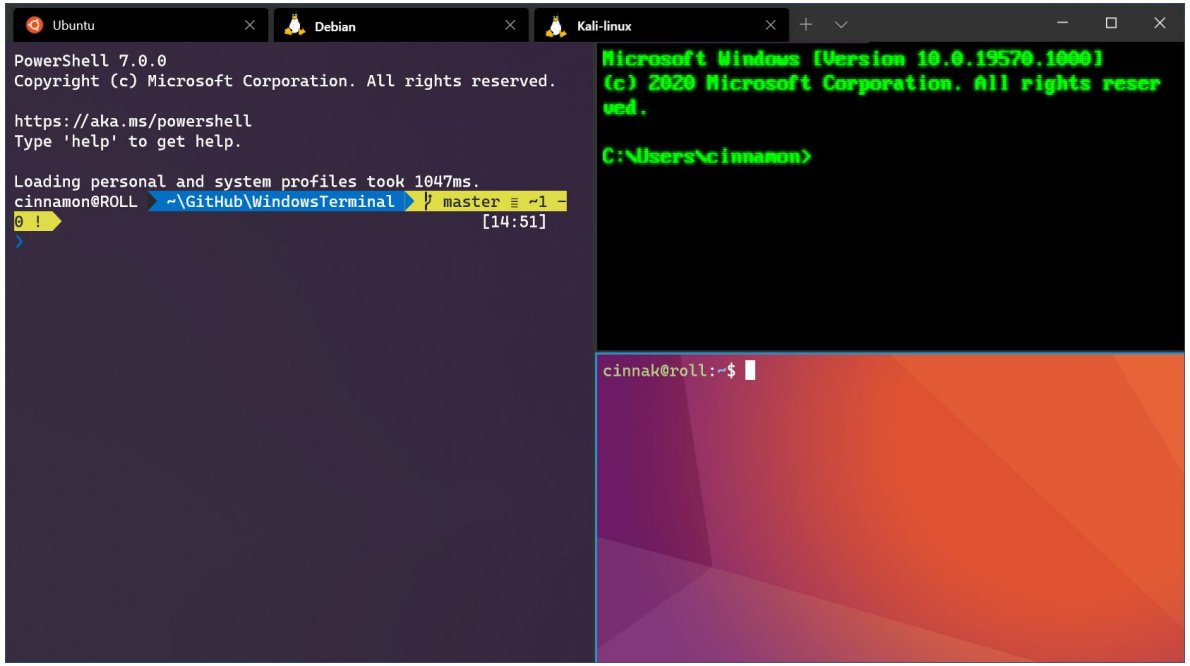
¿Que es una consola de comandos?

Es una aplicación que se encuentra disponible en todos los sistemas operativos y permite interactuar al usuario con el ordenador sin la necesidad de interfaces gráficas. Las UI son herramientas que mejoran la experiencia de usuario; pero internamente se siguen ejecutando comandos de terminal.



¿Por qué utilizar una consola?

Permite obtener un conocimiento profundo del funcionamiento de los sistemas operativos. Además permite gestionar permisos entre los usuarios y archivos. Para usuarios avanzados el uso de la terminal permite automatización de operaciones a través de scripts y el acceso a un mayor control sobre nuestras acciones



Comandos en terminal de windows

- mkdir
- ls
- pwd
- systeminfo
- help
- del
- ipconfig

Documentacion: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands

Comandos en terminal de linux

- ls
- mkdir
- pwd
- help
- ifconfig
- touch
- nano
- echo
- poweroff

Documentación: https://unixguide.readthedocs.io/en/latest/unixcheatsheet/

Ejemplo práctico

Desarrollar un script de terminal en ubuntu que permita el ingreso de dos números e imprima la suma

```
#!/bin/bash
sleep 1
read -p “Ingrese el primer numero” n1
read -p “Ingrese el segundo numero” n2
…
```

Lenguaje de programación C

Programación de microcontroladores ARM - Sesión 1

Hola mundo en C

En este primer programa evaluaremos la forma de programar en C, así como el uso de variables y las herramientas de compilación

Generación de un archivo ejecutable

Para poder ejecutar un archivo en C se debe ejecutar la instrucción "Make" para compilar el archivo en C y obtener el archivo ejecutable. Sin embargo para que el proceso sea más entendible se utilizará la instrucción.

gcc -o holaMundo holaMundo.c

Etapas de compilación

Las etapas de generación del archivo ejecutable son las siguientes:

- Pre-processing
- Compilación
- Ensamblaje
- Enlazamiento

```
> nm execute

000000000040039c r __abi_tag
0000000000401161 T add
0000000000404030 B __bss_start
0000000000404030 b completed.0
0000000000404020 D __data_start
0000000000404020 W data_start
0000000000401070 t deregister_tm_clones
00000000004010e0 t __do_global_dtors_aux
0000000000403e08 d __do_global_dtors_aux_fini_array_entry
0000000000404028 D __dso_handle
0000000000403e10 d _DYNAMIC
0000000000404030 D _edata
0000000000404038 B _end
00000000004011e8 T _fini
0000000000401110 t frame_dummy
0000000000403e00 d __frame_dummy_init_array_entry
0000000000402154 r __FRAME_END__
0000000000404000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
```

Tipos de variables en C

En C se tiene los siguientes tipos de datos:

- char - 1 Byte
- short - 2 bytes
- int - 4 Bytes
- long - 8 Bytes
- float - 4 Bytes
- double - 8 Bytes

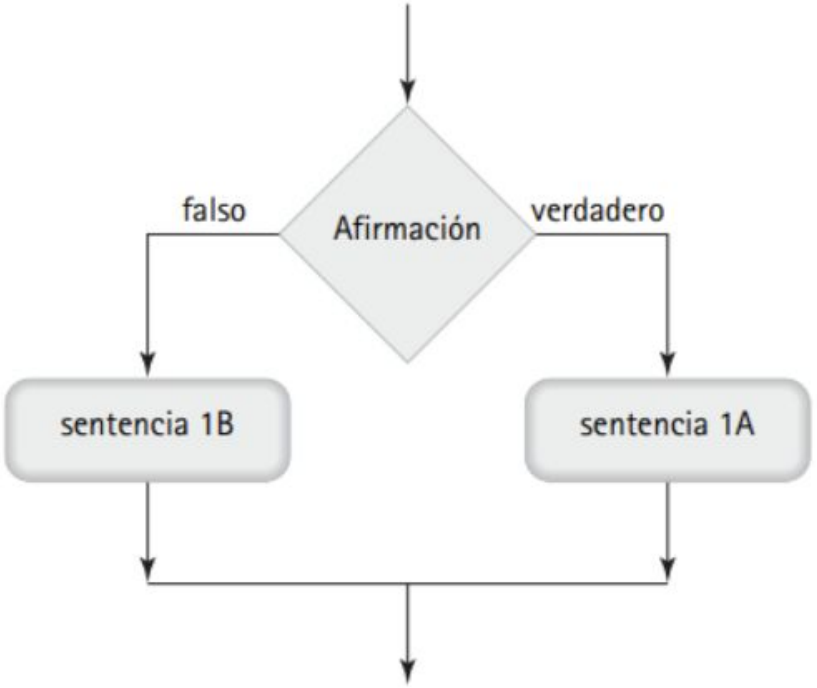
```
int Numero = -20;
int Edad = 15;
unsigned int num = 230;
long var = -2356854;
unsigned long distancia = 48526324;
float peso = 62.50;
char character = 'a';
```

Operadores aritméticos

Operador	Nombre	Ejemplo	Resultado
+	Suma	10 + 5	15
-	Sustracción	10 - 5	5
*	Multiplicación	2*2	4
/	Division	5.0/2.0	2.5
%	Modulo	10%3	1

Sentencias selectivas

Las sentencias selectivas comprueban la veracidad o falsedad de una declaración booleana y ejecutan líneas de código dependiendo del resultado



Operadores booleanos

Los operadores booleanos dan como resultado un valor booleano y son utilizadas en sentencias selectivas e iterativas

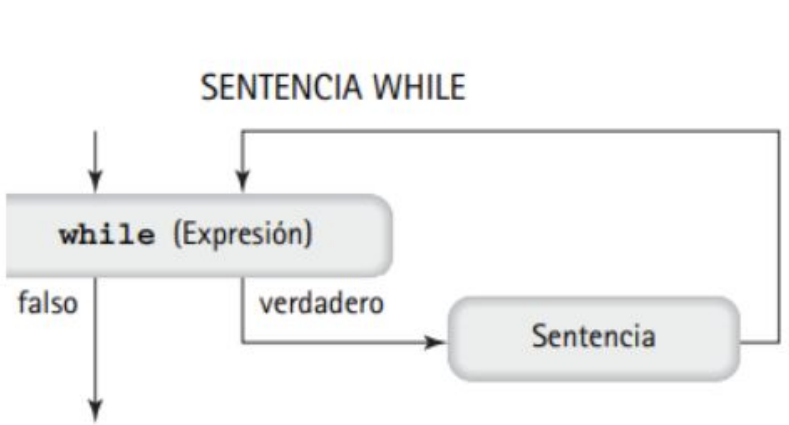
Operadores lógicos y relacionales

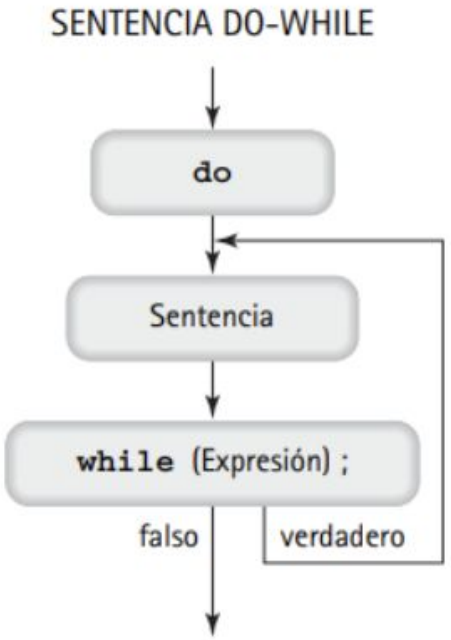
Operador	Nombre	Ejemplo	Resultado
&&	And		
||	Or		
!	Negación		
==	Es igual a	3 == 3 3 == 4	True False
!=	No es igual a	3 != 4	True
<	Es menor que	2 < 1	False
<=	Es menor o igual	4 <= 4	True
>	Es mayor que	3 > 2	True
>=	Es mayor o igual que	5 >= 4	False

Sentencias iterativas

Las sentencias iterativas permiten recorrer ciclos o bucles a través de la comprobación de una sentencia booleana. Las principales sentencias utilizadas para bucles son:

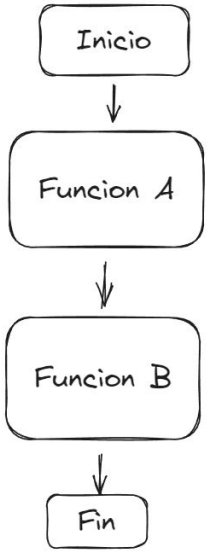
- for
- while
- do-while

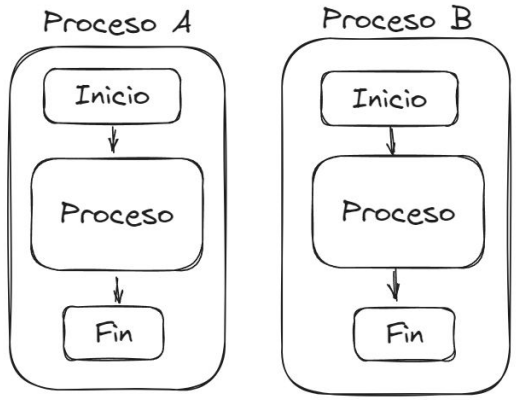




Funciones y procedimientos en C

Las funciones y métodos permiten la modularidad y reutilización en el código C por lo que se recomienda su uso. La declaración de funciones debe realizarse de forma adecuada para que el compilador no produzca errores.





Arreglos y Vectores en C

Los vectores en C se definen como una colección de datos de un tipo en específico, el cual es declarado al momento de inicializar un arreglo.

- <tipoDato> <nombre>[<cantidad>]
- int arregloNumeros[50];

```
/* Declaración de un array. */

#include

main() /* Rellenamos del 0 - 9 */
{
    int vector[10],i;
    for (i=0;i<10;i++) vector[i]=i;
    for (i=0;i<10;i++) printf(" %d",vector[i]);
}
```

Punteros en C

Los punteros en C permiten almacenar direcciones de memoria permitiendo modificar los parámetros por referencia en el llamado de funciones.

Para la utilización de esta estructura se requiere los siguientes operadores:

- * - Apunta a …
- & - Dirección de …