2do Cuatrimestre

- Parcial.
- Formalicemos Funciones.
- Formalicemos Tipos de Datos.
- Formalicemos Sistemas de Numeración.

Funciones

- Metodología top-down
  - Dividir el problema en partes más chicas fáciles de resolver.
- Construir a partir de lo ya hecho.
- Independencia de códigos y datos.
- Formar parte de distintos archivos fuentes y compilarse juntos.
- Le damos “argumentos” y nos devuelve un “único” resultado, el resto es transparente a nuestro código.

Definición Vs. Declaración.

- **Definición**: es la función en sí, contendrá el código que ejecuta dicha función. Una sola vez.
- **Declaración**: Le informa al compilador el formato de la función. Cuando sea necesario
  - Obligatorio: cuando se llama a la función antes de definirla.
  - Obligatorio: cuando se llama a la función en un archivo distinto al de la definición.
  - Optativo: cuando la función es definida antes de su primer llamado.

Definición

- Tres componentes principales:
  a. Primera línea.
  b. Declaración de argumentos.
  c. Cuerpo

No son todas obligatorias.

Definición: primera línea

- Contiene:
  a. Tipo del valor de retorno.
  b. Nombre de la función.
  c. Argumentos que recibe.

  tipo nombre (arg1, arg2,....., argn)
- Los argumentos son pasados por valor (vía pila). Estos no son reconocidos fuera de la función.

Definición: declaración de argumentos

- ➔ Cada tipo de argumento que recibe debe coincidir con el tipo de dato que se le envía desde la llamada a la función
- ➔ Puede ser omitido si se declaran los tipos junto con los argumentos en la primera línea
  a. tipo nombre(arg1, arg2,.....)
     tipo arg1
     tipo arg2
  b. tipo nombre(tipo arg1, tipo arg2,.....)

Definición: cuerpo

- Sentencia compuesta que define las acciones que debe realizar la función.
- Está encerrada entre llaves { }.
- Se utiliza la sentencia return para especificar el valor devuelto por la función.

Declaración

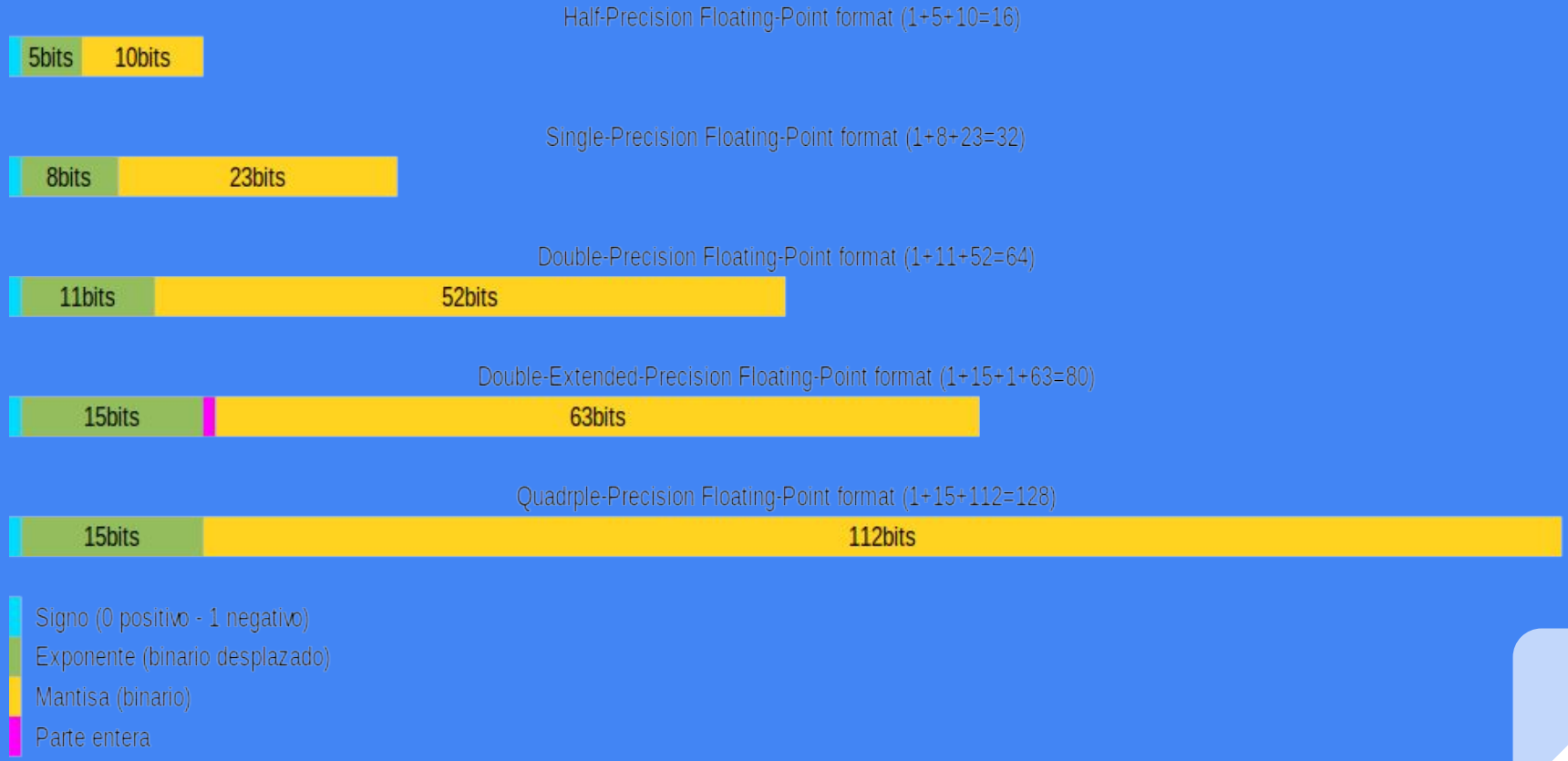
- La idea es darle al compilador la información del formato de la función.
- Es necesario declarar una función cuando:
  - Se la llama antes de su definición
  - Se la llama desde códigos escritos en distintos archivos.
- El formato de una declaración es:
  tipo-almacenamiento tipo-devuelto nombre (tipos-argumentos)

TIPOS DE DATOS

- Carácter: char
- Enteros: int
- Punto flotante simple densidad: float
  - 

- Punto flotante doble densidad: double
  - 

Punto Flotante: IEEE 754



CALIFICADORES

- short
- long
- signed
- unsigned

CONSTANTES

- #define
  - #define ENTERO 1234
  - #define N_LONG 1234L
  - #define N_UNSIGNED 1234U
  - #define CARACTER 'A'
  - #define NVLIN '\n'
  - #define N_HEXA 0x1234
  - #define N_OCTAL 01234
  - #define CADENA “HOLA MUNDO\n”

caracteres especiales

- \a alerta (beep)
- \b backspace
- \f Nueva Hoja
- \n Nueva Línea
- \r Retorno de carro (formfeed)
- \t Tab horizontal
- \v Tab Vertical
- \\ backslash
- \? Signo de pregunta
- \’ Comilla Simple
- \“ Comilla doble
- \ooo Número octal
- \xhh Número Hexadecimal
- \0 nulo

CONSTANTES

- enum
  - enum boolean {NO, SI }; / / NO=0 , SI =1.
  - enum meses {Enero =1 , Febrero , Marzo , A b r i l , Mayo , Junio , J u l i o , Agosto , Setiembre , Octubre , Noviembre , Diciembre }; / / Febrero =2 , Marzo= 3 , . . .
  - enum escapes {BELL = '\a' , TAB = '\t' , NVLIN = '\n'};

Crean un “tipo” de dato constante enum boolean, enum meses y enum escapes respectivamente.

VARIABLES

- **Definición**: es donde se le asigna un lugar en memoria.
  - float a;
  - int b, c; //fuera de la función => externa
  - char texto[100];
- **Declaración**: Informa al compilador la naturaleza de la variable, no asigna memoria; no siempre necesario.
  - float a;
  - extern int b, c; //múltiples archivos
  - char texto[ ];

VARIABLES

- **Inicialización**: es darles un valor inicial en la definición, no es obligatorio.
  - float a=3.14;
  - char texto[11]= “hola mundo”; //10+carácter nulo
  - char texto[]=”hola mundo”

NOTA: son definiciones, no declaraciones.

Tipos de almacenamiento de variables

- Tipos de datos
  - Char, int, float y double.
  - Short, long, signed, unsigned
- Tipos de almacenamiento
  - Según alcance
    - Automáticas (o locales)
    - Externas (o globales)
  - Según duración
    - Estáticas
    - Dinámicas
  - Según velocidad de acceso
    - Por registro
    - En memoria

Almacenamiento automático (auto)

- Se declaran dentro de la función y son locales a la misma.
  - Solo se pueden acceder desde la función
  - No es necesario usar la palabra auto para definirlas

Almacenamiento externo (extern)

- Su ámbito se extiende desde el punto de definición hasta el resto del programa.
  - Mantienen su valor al entrar y salir de las funciones
  - Se define fuera de la función donde puede inicializarse también
  - Solo debe existir una definición de cada variable externa
  - Se definen como externas por su lugar fuera de las funciones por lo que no se usa extern para definirlas.
  - Se la declara usando extern.
  - **No abusar**

Almacenamiento estático (static)

- Aplicable tanto al almacenamiento local como global
  - Estática automática: Solo existen en el ámbito de la función pero mantienen su último valor.
  - Estáticas externas: Sólo serán reconocidas en el archivo donde se las definió.

Almacenamiento en registros (register)

- Le dice al compilador que en lo posible utilice un registro interno del procesador en lugar de la memoria para almacenar su valor.
  - Incrementa la velocidad de acceso
  - Usado cuando la variable tiene un alto grado de aparición en el código.
  - Solo para variables automáticas y parámetros formales de una función.

Resumen

Almacenamiento	Palabra clave	Definición	Declaración	Ámbito (dese su creación)	Vida
Automático o local	auto	Por defecto dentro de una función		Local a la función	De la función
Estático interno	static	**static** dentro de la función		Local a la función	Del programa
Estático externo	static	**static** fuera de la función		Archivo	Del programa
Externo o global	extern	Por defecto fuera de las funciones (1)		Desde su definición	Del programa
Registro	register	**register** dentro de la función		Local a la función	De la función

- Las variables globales y funciones pueden ser definidas como estáticas
- Pueden declararse variables o funciones externas al archivo de definición con “extern”. (1)
- Las definiciones o declaraciones locales pueden ser automáticas, estáticas, externas o registro.

Aclaración

- También existe la posibilidad de definir los tipo:
  - **const**: define una constante en memoria (puede optimizar su acceso estando en memoria de programa).
  - **volatile**: define una variable que puede ser modificada fuera del programa para obligar a la lectura de la memoria siempre que se la utilice.

Representación BINARIA

- ➔ Esta correspondencia entre señal DIGITAL y numeración BINARIA hacen a esta última una herramienta ideal para representar y analizar el comportamiento de las primeras.
- ➔ La máquina usa niveles de tensión y el hombre los representa con números binarios.

Sistema DECIMAL

- El sistema decimal utiliza una BASE=10
  - Necesito 10 símbolos (0 1 2 3 4 5 6 7 8 9).
  - El valor del número viene dado por la suma de cada dígito multiplicado por su “peso”.

Por ejemplo:

$$123{,}4 = 1 \cdot 10^{2} + 2 \cdot 10^{1} + 3 \cdot 10^{0} + 4 \cdot 10^{-1}$$

Sistema BINARIO

- El sistema binario utiliza una BASE=2
  - Necesito 2 símbolos (0 1).
  - El valor del número viene dado por la suma de cada dígito multiplicado por su “peso”.

Por ejemplo:

$$10110_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 16 + 0 + 4 + 2 + 0 = 22_{10}$$

Sistema OCTAL

- El sistema octal utiliza un BASE=8
  - Necesito 8 símbolos (0 1 2 3 4 5 6 7).
  - El valor del número viene dado por la suma de cada dígito multiplicado por su “peso”.

Por ejemplo:

$$377_8 = 3 \cdot 8^2 + 7 \cdot 8^1 + 7 \cdot 8^0 = 192 + 56 + 7 = 255_{10} \quad (255 = 4 \cdot 8^2 - 1)$$

Sistema HEXADECIMAL

- El sistema hexadecimal utiliza un BASE=16
  - Necesito 8 símbolos (0 1 2 3 4 5 6 7 8 9 A B C D E F).
  - El valor del número viene dado por la suma de cada dígito multiplicado por su “peso”.

Por ejemplo:

$$ABC_{16} = 10 \cdot 16^2 + 11 \cdot 16^1 + 12 \cdot 16^0 = 2560 + 176 + 12 = 2748_{10}$$

Que quede claro

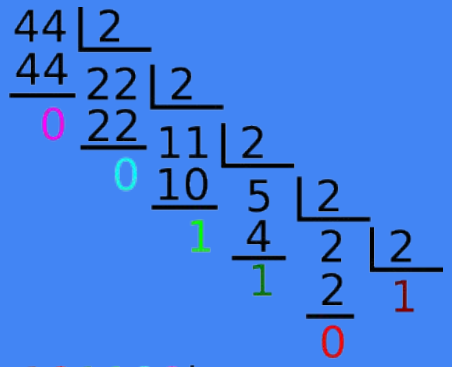
- ➔ Un número lo podemos representar en cualquier base.
- ➔ Al representarlo en distintas bases cambia su expresión simbólica pero no su magnitud.

A5h=165d=245o=10100101b

Métodos para CAMBIO de BASE

Divisiones Sucesivas

Se divide el número a convertir por la base a convertir, hasta que el cociente de un número menor que dicha base. El resultado se compone del último cociente y los restos tomados en sentido inverso a la sucesión de cocientes.



44d=101100b

Métodos para CAMBIO de BASE

Divisiones Sucesivas

Se divide el número a convertir por la base a convertir, hasta que el cociente de un número menor que dicha base. El resultado se compone del último cociente y los restos tomados en sentido inverso a la sucesión de cocientes.

En el caso de los decimales se multiplica por la base y la parte entera es el dígito buscado.

0,375x2=**0**,75 => 0,**0**

0,75x2=**1**,5 => 0,0**1**

0,5x2=**1** => 0,011

2,375d=10,011b

Métodos para CAMBIO de BASE

Restas sucesivas

Consiste en tomar el número a convertir y buscar la potencia de 2 más grande que se le pueda restar, tomando el resultado de la resta como nuevo número para seguir el proceso, siempre que este no sea negativo.

$44-2^5$	12
$12-2^3$	4
$4-2^2$	0

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	1	0	0

Conversiones Rápidas

- Binario a Octal.

1	1	0	0	1	0	1	1	1	0
1		4			5			6	

- Binario a Hexadecimal.

1	1	0	0	1	0	1	1	1	0
3			2			E			

1100101110b=1456o=32Eh

TABLA

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12

Representación Binaria en la PC

- Las PCs, como otros dispositivos digitales, solo trabajan con dos niveles (0 y 1)
- Problemas:
  - ¿Cómo representar números negativos?
  - ¿Cómo representar números con coma?
  - Estamos limitados en el ancho de los números que podemos manejar dentro de la PC.

Sistema Binario: UNIDADES

- Bit [b]: acrónimo de Binary digit, (0 o 1).
- Nibble: Conjunto de 4 bits.
- Byte [B]: Conjunto de 8 bits.
- Word [w]: Conjunto de 16 bits.
- DoubleWord: Conjunto de 32 bits.
- QuadWord: Conjunto de 64 bits.

Suma BINARIA

- Mientras que la suma no supere el tamaño máximo soportado, la suma se efectúa sin problema.
- Pero si supero el máximo número de dígitos que manejo tendré problemas.
  - Esto se llama CARRY (flag c)

```
   111
  10011101b
+ 01001100b
-----------
  11101001b
```

```
 11 1
  11010110b
+ 01010000b
-----------
 ?00100110b
```

Operaciones Básicas

Sumar y restar dentro del conjunto de los ENTEROS nos trae un nuevo problema a la hora de hacerlo en una PC:

- ¿Cómo representar números negativos?
- Analizaremos distintos métodos:
- Signo y Magnitud.
- Complemento a 1.
- Complemento a 2.
- Binario Desplazado.

Signo y Magnitud

- Como ya mencionamos, dentro de la PC solo podemos representar 0s y 1s.
- Signo y Magnitud utiliza el bit más significativo (MSB) para representar el signo y el resto para el módulo.
- Así, trabajando con 4 bits tenemos que:

0010b equivale a 2.

1010b equivale a -2.

Signo y Magnitud

Este método tiene algunos inconveniente:

- ➔ Se consumen 2 valores para representar al cero.

  0000b=1000b (0=-0)

- ➔ No se puede usar el mismo HARDWARE que suma números positivos para sumar números negativos.

0010b+1001b=1011b (-3)

2+(-1)=1

Complemento a 1 (C1)

- Para representar un número negativo se invierte cada bit por su complemento (1 en 0 y viceversa).

  0111b=7d y 1000b=-0111b=-7d

- Continuamos teniendo el problema de ocupar dos números para representar al 0.

  0000b=0d y 1111b=-0000b=-0d

- Al sumar dos números de distinto signo hay que sumar el acarreo del MSB para no tener errores.
- Esto significa que si bien podremos usar el mismo hardware este tendrá que analizar de qué caso se trata.

OPERACIÓN SUMA

- Si ocurre un carry al final de la adición/resta, sumarlo al resultado obtenido (end-around carry)
  - 0010b+0100b=0110b (2+4=6)
  - 0010b+1110b=^{1}0000b+1=1 (2+(-1)=1)
  - 0010b+1100b=1110b=-1 (2+(-3)=-1)
  - 1101b+1011b=^{1}1000b+1=-6 (-2+(-4)=-6)
  - 0110b+0110b=1100b=-3 (6+6=12 => fuera de escala)
  - 1001b+1001b=^{1}0010b+1=3 (-6-6=-12 => fuera de escala)

¡¡¡ Cuidado con salirse de escala !!!

NOTA

Los protocolos de Internet IPv4, ICMP, UDP y TCP usan todos el mismo algoritmo de suma de verificación de 16 bits en complemento a uno. Aunque la mayoría de la computadoras carecen del hardware para manejar acarreo del último bit (end-around carry), la complejidad adicional es aceptada ya que es igualmente sensible a errores en todas las posiciones de bits. En UDP, una representación de todos ceros indica que la suma de verificación opcional ha sido omitida. La otra representación, todos unos, indica un valor 0 en la suma de verificación (las sumas de verificación son obligatorias para IPv4, TCP e ICMP; fueron omitidas en IPv6).

Complemento a 2 (C2)

- Para representar un número negativo se invierte cada bit por su complemento y se le suma 1 (C2=C1+1).

0111b=7d
0000b=0d
1111b = -0000b+1= -1d
1000b = -0111b+1= -8d

- Se elimina la ambigüedad del cero.
- No requiere ajuste al sumar 2 números de distinto signo.
- Podremos usar el mismo hardware.

OPERACIÓN SUMA

- Si ocurre un carry al final de la adición/resta, sumarlo al resultado obtenido (end-around carry)
  - 0010b+0100b=0110b (2+4=6)
  - 0010b+1111b=^{1}0001b=1 (2+(-1)=1)
  - 0010b+1101b=1111b=-1 (2+(-3)=-1)
  - 1110b+1100b=^{1}1010b=-6 (-2+(-4)=-6)
  - 0110b+0110b=1100b=-4 (6+6=12 => fuera de escala)
  - 1010b+1010b=^{1}0100b=4 (-6-6=-12 => fuera de escala)

¡¡¡ Cuidado con salirse de escala !!!

Binario Desplazado

- Se suma al valor signado el valor absoluto de la mitad del rango menos 1.
- El formato en exceso es habitual para la representación del exponente en números en punto flotante
- Ejemplos en 3bits:

$$-3+\frac{2^3}{2}-1=-3+4-1=0\Rightarrow -3\mathrm{d}=000\mathrm{b}$$

$$0+\frac{2^3}{2}-1=0+4-1=3\Rightarrow 0\mathrm{d}=011\mathrm{b}$$

$$4+\frac{2^3}{2}-1=4+4-1=7\Rightarrow 7\mathrm{d}=111\mathrm{b}$$

Resumen para 4 bits				
Decimal	Bit de Signo	Complemento a 1	Complemento a 2	Binario desplazado
8				1111
7	0111	0111	0111	1110
6	0110	0110	0110	1101
5	0101	0101	0101	1100
4	0100	0100	0100	1011
3	0011	0011	0011	1010
2	0010	0010	0010	1001
1	0001	0001	0001	1000
(+)0	0000	0000	0000	0111
(-)0	1000	1111		
-1	1001	1110	1111	0110
-2	1010	1101	1110	0101
-3	1011	1100	1101	0100
-4	1100	1011	1100	0011
-5	1101	1010	1011	0010
-6	1110	1001	1010	0001
-7	1111	1000	1001	0000
-8			1000	

FLAGs aritmeticos

- Z = Zero flag: El resultado es cero.
- N = Negative flag: El resultado es negativo.
- V = Overflow flag: El resultado supera el número de bits que puede manejar la ALU.
- P = Parity flag: Paridad del número de 1 en los datos.
- C = Carry flag: Acarreo de la operación realizada.

Representar REALES

- ➔ Por ahora solo vimos como representar un pequeño subconjunto de los enteros.
- ➔ La codificación en PUNTO FLOTANTE permite trabajar (en un mismo tipo de dato) con magnitudes muy grandes y muy pequeñas.
- ➔ Pero el precio a pagar: a mayor rango menor será la precisión y viceversa.

RANGO y PRECISIÓN

- Se divide el número de bits en tres:
  - Signo
  - Mantisa (determina el número de dígitos que podemos representar: precisión)
  - Exponente (determina el rango de números que podemos representar)
- Tener el número de dígitos fijos me limita los números que puedo representar.
- Por ejemplo, con 4 dígitos decimales tendremos:
  - Rango:
    - 0000 a 9999
    - 0.000 a 9.999
  - Precisión:
    - Del 1000 paso al 1001 (no puedo representar el 1000.1)
    - Del 1.001 paso al 1.002 (no puedo representar el 1.0015)
-

Representación Binaria de Números Reales

- En general tenemos dos formatos:
  - Punto Fijo.
  - Punto Flotante.

Punto Fijo con Signo

➔ Se representan mediante una expresion del tipo:

$(a_n\ a_{n-1} \cdots a_0 . a_{-1}\ a_{-2} \cdots a_{-m})_b = (-1)^s * (a_n * 2^n + \cdots + a_0 * 2^0 + a_{-1} * 2^{-1} + a_{-2} * 2^{-2} + \cdots + a_{-m} * 2^{-m})$

- s es el signo: 0 positivo y 1 negativo.
- La distancia entre dos números consecutivos es 2-m.
- Ejemplo: $-101.011b=(-1)^1[1x2^2+0x2^1+1x2^0+0x2^{-1}+1x2^{-2}+1x2^{-3}]$

Deja de ser un rango continuo de números para transformarse en un rango discreto. Por ejemplo, incrementos de 0.001b=0.125d

Punto Flotante: Notación Científica

Sistema Decimal

- Permite representar números reales.
- Su formato es: $\pm a \cdot 10^{b}$ $\left(otra\ forma \pm aEb\right)$
  - a: coeficiente
  - b: exponente
- Ejemplo: $1234 \cdot 10^{-2} = 12{,}34$
- Para unificar la representación se utiliza la Notación Científica Normalizada:

$$\pm a \cdot 10^{b} \quad con \quad 0{,}1 \leqslant a < 1 \wedge b \in \mathbb{Z}$$

Punto Flotante: Representación

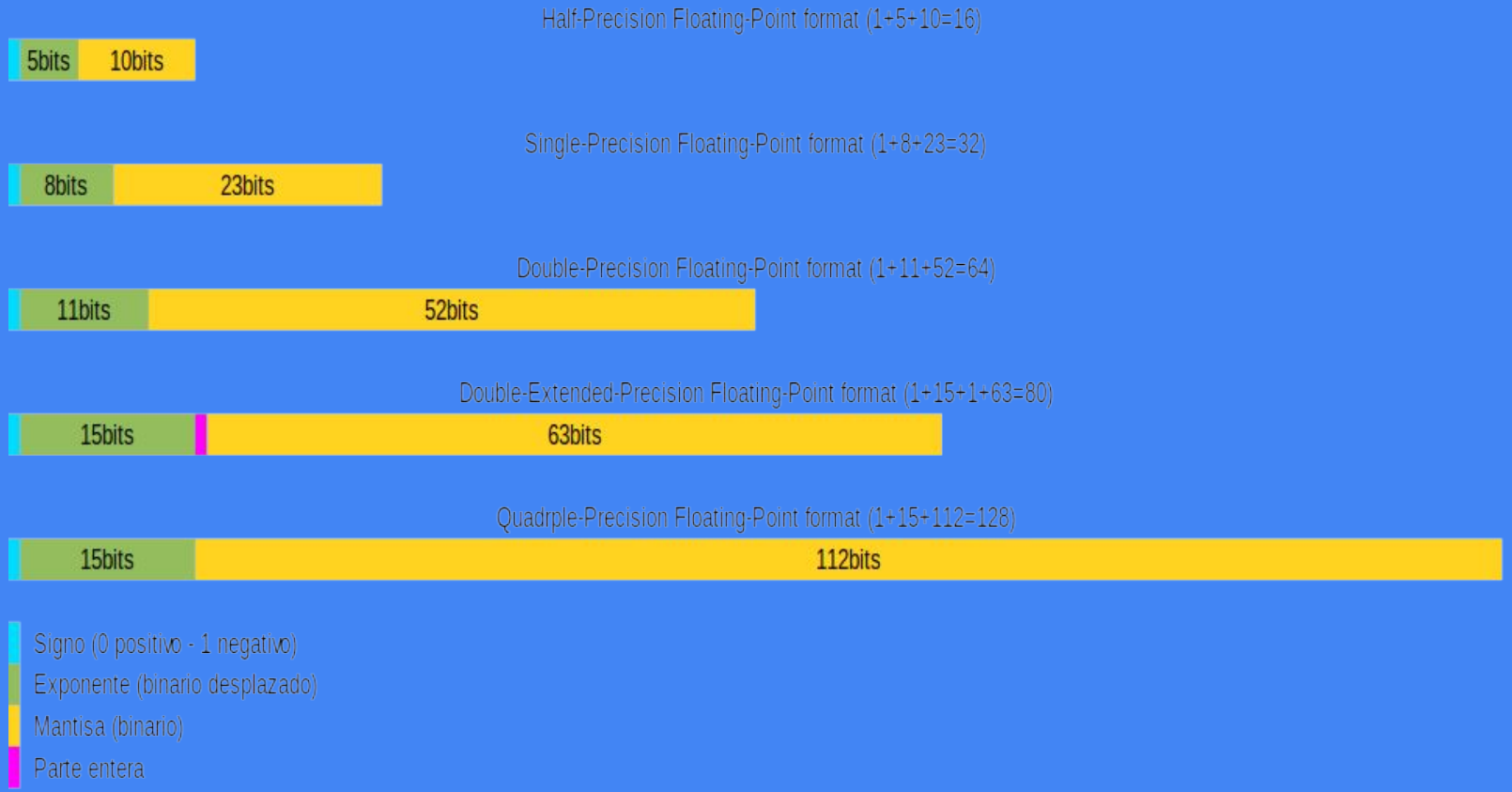
- Se representan con los pares ordenados (m,e): $(m, e) = m \cdot b^e$

  m: mantisa; representa a un número fraccionario.
  b: base; toma el valor del sistema de numeración.
  e: exponente; un número entero.

Punto Flotante: Formato IEEE 754

- IEEE: Institute of Electrical and Electronic Engineers
  - Asociación técnico-profesional mundial dedicada a la estandarización.
- IEEE 754
  - Estándar para aritmética en coma flotante.
  - Ampliamente utilizado.
  - Define representación, operaciones y valores especiales.
  - Formatos:
    - 16bits (Half precision)
    - 32bits (Single precision)
    - 64bits (Double precision)
    - 80bits (Double Extended Precision)
    - 128bits (Quadruple precision)
-

Punto Flotante: FORMATOS



Punto Flotante: RANGOS

Formato	Min. Desnormalizado	Min. Normalizado	Max. Finito	Dig. Dec.
Half	≈ 5.96E−8	≈ 6.1E−5	65504	≈ 3,119
Single	≈ 1.4E−45	≈ 1.18E−38	≈ 3.4E38	≈ 7,225
Double	≈ 4.9E-324	≈ 2.2E-308	≈ 1.8E308	≈ 15,955
Extended	≈ 3.6E-4951	≈ 3.4E-4932	≈ 1.2E4932	≈ 19,266
Quadruple	≈ 6.5E-4966	≈ 3.4E-4932	≈ 9.63E-35	≈ 34,016

Representación de caracteres

- No solo necesitaremos representar números.
- Para representar letras y símbolos se utilizan distintos tipos de codificaciones.
- Su representación será, sin duda, binaria; pero tendrá que estar codificada por algún estándar.
  - ASCII
  - ISO 8859-1
  - Unicode
  - UTF

ASCII

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
- Creado en 1963 por el Comité Estadounidense de estándares ASA (ANSI desde 1969)
- Define 95 caracteres imprimibles, numerados del 32 al 126 (del 0 al 31 y el 127 son caracteres no imprimibles).
- Basado en el alfabeto LATINO.
- Utiliza 7 bits para representar los caracteres (1111111b=127).
- ASCII extendido: utiliza los 8bits

Tabla ASCII ESTANDAR

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Tabla ASCII EXTENDIDA

➔

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	Ö	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	