2010

UNAN – LEON
Departamento de Computación
Ing. En Sistemas de Información

Autor: Ing.Karina Esquivel Alvarado.

Asignatura: Programación II.

INTRODUCCIÓN AL LENGUAJE C

TEMA 1: INTRODUCCIÓN AL LENGUAJE C

1.1 INTRODUCCIÓN:

Una computadora es una máquina para procesar información y obtener resultados en función de unos datos de entrada.

- **Hardware:** parte física de una computadora (dispositivos electrónicos).
- **Software:** parte lógica de una computadora.

Las computadoras se componen de:

- Dispositivos de Entrada/Salida.
- Unidad Central de Proceso (Unidad de Control Memoria central).
- Dispositivos de almacenamiento masivo de información (Memoria Auxiliar o Externa).

El software del sistema comprende, entro otros, el sistema operativo MSDOS, UNIX, Linux... y los lenguajes de Programación.

Los lenguajes de programación se clasifican en:
Alto nivel: Pascal, FORTRAN, VISUAL, BASIC, C, Ada, Modula-2, Ctt-, Java, Delphi, C, etc.
Bajo nivel: Ensamblador.
Máquina: Código máquina.

C es un lenguaje de **alto nivel,** que permite programar con instrucciones de lenguaje de propósito general. También, C se define como un lenguaje de programación estructurado de propósito general; que proporciona potencia y flexibilidad al desarrollo de sistemas. Hoy, en el siglo XXI, C sigue siendo uno de los lenguajes de programación más utilizados en la industria del software, así como en institutos tecnológicos, escuelas de ingeniería y universidades.

1.2 ESTRUCTURA GENERAL DE UN PROGRAMA EN C:

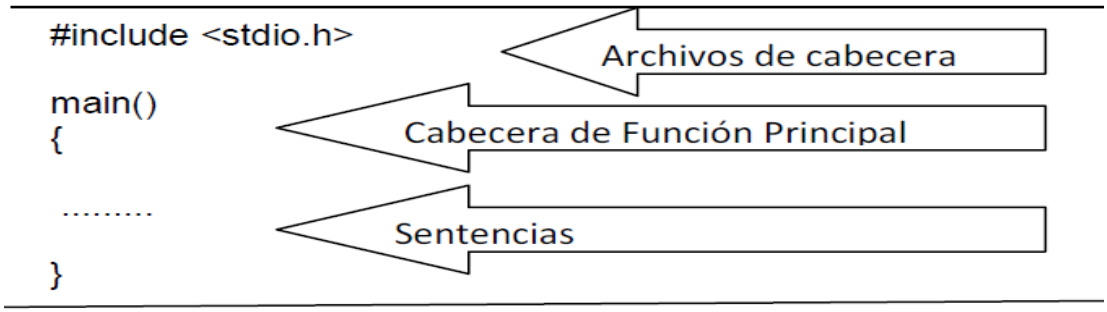
Un programa en C es una estructura que se compone de una o más funciones, una función que tiene que ser obligatoria es la **función main( )** esta es conocida como la función principal.

• El programa comienza con la función: main( ).

• Cada función o programa, consta de un cuerpo de función delimitado por llaves de comienzo y fin { }.

• En el cuerpo de la función irán incluidas: sentencias, variables, funciones, etc. terminadas cada una de ellas por un punto y coma ";"

Así:



De un modo más explícito, un programa C puede incluir:

- Directivas de preprocesador;
- Declaraciones globales;
- La función main ( );
- Funciones definidas por el usuario;
- Comentarios del programa (utilizados en su totalidad). // o /* */

Ejemplo#1: Programa en C que imprime un Mensaje.

```
#include<stdio.h>
main( )
{
        printf("Bienvenido a la programación en C");
}
```

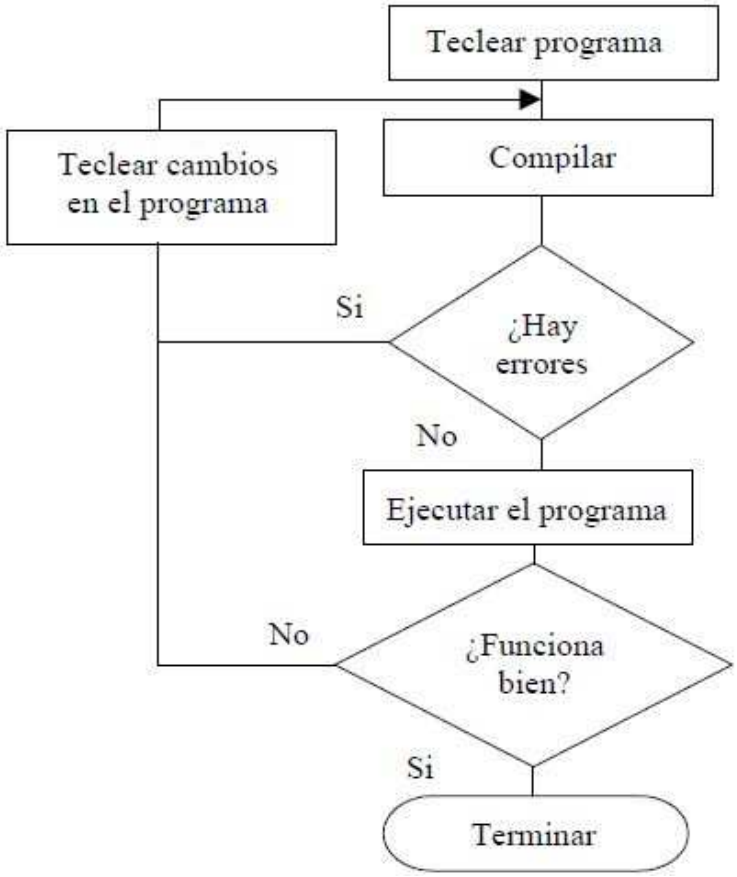
La directiva #include de la **primera línea** es necesaria para que el programa tenga salida. Se refiere a un archivo externo denominado stdio.h en el que se proporciona la información relativa a la función printf( ) . Obsérvese que los ángulos < y > no son parte del nombre del archivo; se utilizan para indicar que el archivo es un archivo de la biblioteca estándar C.
La **segunda línea** contiene la cabecera de la función main( ) obligatoria en cada programa C. Indica el comienzo del programa y requieren los paréntesis ( ) a continuación de main ( ) .
La **tercera y quinta** línea contienen sólo las llaves { y } que encierran el cuerpo de la función main ( ) y son necesarias en todos los programas C.
La **cuarta línea** contiene la sentencia printf("Bienvenido a la programación en C"); que indica al sistema que escriba el mensaje "Bienvenido a la programación en C" .

La salida será:

- **Instrucciones (Sentencias):** Una instrucción es el elemento básico de cualquier lenguaje, con el le expresamos al compilador cual es nuestro deseo. Las instrucciones son simplemente órdenes, y tenemos a nuestra disposición; instrucciones de control, llamadas, impresiones, definir un dato o archivo. Todo lo haremos con instrucciones.
- **Función main():** Es parte esencial de un programa para el lenguaje C, ya que la función main se encuentra dentro del encabezado del programa. Es también allí donde tiene inicio el programa.

1.3 ¿CÓMO SE CREA UN PROGRAMA EN C?

Los pasos necesarios para desarrollar un programa C son los siguientes:
a) Edición: Utilizar un editor para escribir el programa fuente texto.
b) Compilación: Compilar el programa fuente, es decir, traducir el programa a lenguaje máquina.
c) Ejecución: Una vez compilado se procede a la ejecución del programa tecleando el nombre del fichero-programa.



1.4 ELEMENTOS DE UN PROGRAMA EN C:

1.4.1 Identificadores:

Un identificador es una secuencia de caracteres, letras, dígitos y subrayados (_). El primer carácter debe ser una letra (algunos compiladores admiten carácter de subrayado). Las letras minúsculas y mayúsculas son diferentes.

Ejemplos de Identificadores Válidos:
Nombre_de_clase
Linea
Fecha
Elemento_mayor
B
Lunes20
Dirección_1
Nombre_empl
Casa_34

Reglas básicas de formación de identificadores:
1. Secuencia de letras o dígitos; el primer carácter puede ser una letra o un subrayado.
2. Los identificadores son sensibles a las mayúsculas: C reconoce entre mayúscula y minúscula así que decir PI es diferente a Pi o a pi.
3. Los identificadores pueden tener hasta 255 caracteres de longitud.
4. Los identificadores no pueden ser palabras reservadas, tales como if, switch o else.

Nota: Es importante hacer notar que para un mejor trabajo y entendimiento se recomienda que los identificadores se escriban con letras minúsculas.

1.4.2 Palabras Reservadas:

El lenguaje C reserva algunas palabras para el uso de identificadores especiales o bien para denotar algunos elementos, estas palabras reservadas no se pueden utilizar más que para su debido propósito, por ejemplo **void** es una palabra reservada para determinar un tipo especial de datos, así que sólo para eso se utilizará y no le daremos otro uso.

Ejemplo:
```
int main( )
{
        int float;
}
```

Estos son identificadores de palabras reservadas y no se pueden usar para otro propósito:

	enum	signed
auto	extern	sizeof
break	float	static
case	for	struct
char	goto	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	register	void
double	return	volatile
else	short	while

1.4.3 Comentarios en C:

Los comentarios en lenguaje C son de mucha importancia para el programador, estos le ayudan a recordar lo que realiza el programa o bien qué parte del programa realiza una actividad específica, así mismo permite a los programadores que comprendan bien el código para el que lo ha realizado y quiere en el futuro mejorarlo.

En C los comentarios multilínea son de la siguiente forma:
/*Texto que se desea comentar*/

O bien
// .....otro tipo de hacer comentario en C línea a línea.

Para ambos el comentario que se introduzca, no será tomado en cuenta por el compilador.

Ejemplo:
/* Este es el comentario de un programa en C que realiza la suma de los números del 20 al 1000. Recibe solo valores enteros */

1.4.4 Signos de Puntuación:

Las sentencias en C siempre tienen que terminar con; **(punto y coma)** existen en C otros signos de puntuación como:

!	%	^	&	*	(	)	-	+	=	{	}	~
[	]	\	;	'	:	<	>	?	,	.	/	"

1.4.5 Archivos de Cabecera:

Los archivos de cabecera son archivos especiales que contiene librerías de funciones del propio compilador o bien las creadas por el usuario, estos archivos tienen la extensión .h y están puestos en los programas con las directivas **#include** para insertar el archivo correspondiente, por ejemplo si se utiliza en el programa la función para obtener la raíz cuadrada de un número ***sqrt()***, en el programa tiene que llevar la correspondiente directiva ***#include<math.h>***

La mayoría de los programas tienen líneas como la siguiente:
#include<stdio.h> o bien **#include "stdio.h"**, la primera hace al compilador que busque en el subdirectorio actual o por defecto donde se encuentran las librerías y el segundo busca los directorios de trabajo como D: , C: y si no encuentra busca en el directorio actual.

1.5 TIPOS DE DATOS EN C:

El lenguaje C tiene un gran número de datos predefinidos, así mismo brinda la posibilidad para que el usuario cree sus propios tipos de datos. Todos los tipos de datos esencialmente son números.

Los tipos de datos en C se clasifican en: tipos primitivos y tipos derivados.

1.5.1 Tipos de Datos Primitivos:

Los datos manejados en C pueden ser de cinco tipos básicos.

- ✓ **int:** enteros sin decimales entre (-32768 y +32767). Ocupan en la memoria 2 bytes.
- ✓ **float:** números decimales entre (3.4E-38 a 3.4E+38) con ocho dígitos de precisión. Ocupan en la memoria 4 bytes.
- ✓ **char:** caracteres alfabéticos, signos especiales, etc. El rango es (0 y 255). Ocupan en la memoria 1 byte.
- ✓ **double:** números decimales entre (1.7E-308 a 1.7E+308) con 16 dígitos de precisión. Ocupan en la memoria 8 bytes.
- ✓ **void:** sin valor. No almacenar nada y por tanto no necesitan espacio físico de la memoria.

1.5.2 Tipos de Datos Derivados:

Son construidos a partir de los tipos primitivos. Algunos de ellos son: arrays, estructuras, uniones, punteros.

1.5.3 Identificadores de Tipo:

Todos los tipos básicos excepto **void** pueden tener modificadores. Se usan para alterar el significado de un tipo base de acuerdo con nuestras necesidades.
Los tipos de datos primitivos pueden tener distintos modificadores precediéndolos. Un modificador se usa para alterar el significado del tipo primitivo, de forma que se ajuste más precisamente a las necesidades de cada momento.

Los modificadores son:

- ✓ **signed:** entero con signo, es decir un entero positivo o negativo.
- ✓ **unsigned:** entero sin signo, el cual es manipulado como un valor entero positivo.
- ✓ **long:** entero o decimal largo.
- ✓ **short:** entero o decimal corto.

Se pueden aplicar todos los modificadores para los tipos base carácter y entero. También se puede aplicar el modificador ***long a double***. A continuación mostramos todas las posibles combinaciones de los tipos básicos y los modificadores.

Tipo	Bytes	Rango
char	1	-128 a 127
unsigned char	1	0 a 255
signed char	1	-128 a 127
int	2	-32768 a 32767
unsigned int	2	0 a 65535
signed int	2	-32768 a 32767
short int	2	-32768 a 32767
unsigned short int	2	0 a 65535
signed short int	2	-32768 a 32767
long int	4	-2147483648 a 2147483647
signed long int	4	-2147483648 a 2147483647
unsigned long int	4	0 a 4294967295
float	4	3.4E-38 a 3.4E+38
double	8	1.7E-308 a 1.7E+308
long double	8	1.7E-308 a 1.7E+308

NOTA: El espacio ocupado en la memoria por los tipos de datos aquí mostrados vendrá determinado por el tipo de compilador implementado. En cualquier caso, mediante el operador sizeof(tipo de dato) se podrá determinar el número de bytes ocupados.

1.5.4 Variables:

Una variable representa un espacio de memoria para almacenar un valor de un determinado tipo. El valor de una variable, a diferencia de una constante, puede cambiar durante la ejecución de un programa. Las variables deben ser declaradas antes de usarlas.

Sintaxis: tipo_identificador [, identificador]...;
Tipo: Es cualquier tipo de dato válido con algún modificador.

Ejemplo de declaración de variables:

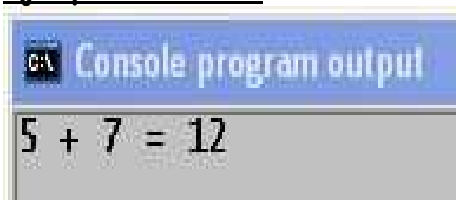
- int i,j,l;
- short int si;
- unsigned int ui;
- double salario, gastos;

Como nombres de variables se admiten letras y números:
• El primer carácter debe ser una letra.
• No se admiten espacios en blanco pero sí el signo de subrayado.
• Mayúsculas y minúsculas se diferencian.
• No se pueden emplear palabras clave del lenguaje.

Ejemplo #2: Programa en C que suma dos números de tipo entero.

```
#include<stdio.h>
void main( )
{
        int valor1=5, valor2=7, resul;
        resul = valor1 + valor2;
        printf("%d + %d = %d\n\n",valor1, valor2,resul);
}
```

Ejemplo de Salida:



Ámbito de la variable: En dependencia del lugar de donde se definen las variables de C, estas se pueden utilizar en la totalidad del programa, dentro de una función o puede existir solo temporalmente dentro de un bloque de una función; la zona donde una variable está activa se denomina ámbito o alcance.

1.5.5 Constantes:

Declarar una constante significa decirle al compilador C el nombre de la constante y su valor. Una vez declarada e iniciada una constante, ya no se puede modificar su valor. Esto se hace antes de la función main( ) utilizando la directriz #define, cuya sintaxis es la siguiente:

#define NOMBRE_CONSTANTE VALOR

Observe que no hay un ; después de la declaración, esto es así, porque una directriz no es una sentencia C, sino una orden para el preprocesador. El tipo de una constante es el tipo del valor asignado. Suele ser habitual escribir el nombre de una constante en mayúscula.

Ejemplos:
#define PI 3.1416
#define NDIAS_SEMANA 7
#define MENSAJE "Pulse una tecla para continuar"

1.6 EXPRESIONES NUMÉRICAS EN C:

Una expresión es un conjunto de operandos unidos mediante operadores para especificar una operación determinada. Todas las expresiones cuando se evalúan retornan un valor.

Tomar en cuenta que:

- ✓ Cuando C tiene que evaluar una expresión en la que intervienen operandos de diferentes tipos, primero convierte, sólo para realizar las operaciones solicitadas, los valores de los operandos al tipo del operando cuya precisión sea más alta (por ejemplo, un int es las preciso que char y un double es más preciso que un int).

- ✓ Cuando se trate de una asignación, convierte el valor de la derecha al tipo de la variable de la izquierda siempre que la conversión se realice explícitamente.

1.7 OPERADORES EN C:

Los operadores son símbolos que indican como manipulados los datos. Se pueden clasificar en:

- ✓ **Operadores Aritméticos.**
- ✓ **Operadores Lógicos.**
- ✓ **Operadores Relacionales.**
- ✓ **Operadores Unitarios.**
- ✓ **Operador Condicional.**

Operadores aritméticos: Utilizados para realizar operaciones aritméticas.

Operador	Operación
+	**Suma.** Los operandos pueden ser enteros o reales.
-	**Resta.** Los operandos pueden ser enteros o reales.
*	**Multiplicación.** Los operandos pueden ser enteros o reales.
/	**División**: Los operandos pueden ser enteros o reales. Si ambos operadores son enteros el resultado es entero. En el resto de los casos el resultado es real segundo?
%	**Módulo o resto de una división entera.** Los operandos tienen que ser enteros.

Operadores de Relación: Comprueban una relación entre dos operandos. El resultado de una operación de relación es un valor booleano (0 o 1).

Operador	Operación
>	¿Primer operando mayor que el segundo?
>=	¿Primer operando mayor o igual que el segundo?
<	¿Primer operando menor que el segundo?
<=	¿Primer operando menor o igual que el segundo?
!=	¿Primer operando distinto que el segundo?
==	¿Primer operando igual que el segundo?

Operadores Lógicos: El resultado de una operación lógica (AND, OR, NOT) es un valor booleano (1 o 0). Las expresiones que dan como resultado valores booleanos pueden combinarse para formar expresiones booleanas utilizando los operadores lógicos.

En C, toda expresión numérica con un valor diferente de 0 se corresponde con un valor verdadero y toda expresión numérica de valor 0, con falso.

Operador	Operación
&&	**AND.** Da como resultado verdadero si al evaluar cada uno de los operandos el resultado es verdadero. Si uno de ellos es falso, el resultado es falso.
||	**OR**. El resultado es falso si al evaluar cada uno de los operandos el resultado es verdadero. Si uno de ellos es verdadero, el resultado es verdadero.
!	**NOT.** El resultado de aplicar este operador es falso si al evaluar su operando el resultado es verdadero, y verdadero en caso contrario.

Operadores Unitarios: Se aplican a un solo operando y son: ++, --.

Operador	Operación
++	Incremento
--	Decremento

Si los operadores ++ y -- están de prefijos, la operación de incremento o decremento se efectúa antes que la operación de asignación; si los operadores ++ y -- están de sufijos, la asignación se efectúa en primer lugar y la incrementación o decrementación a continuación.

Ejemplo:

```
#include<stdio.h>
void main()
{
        int n = 8;
        printf("n++= %d\n", n++);     //Postincremento
        printf("++n = %d\n", ++n);    //Preincremento
        printf("n-- = %d\n", n--);    //Postdecremento
        printf("--n = %d\n", --n);     //Predecremento
}
```

Salida:

Operadores Condicional (?:), llamado también operador ternario, se utiliza en expresiones condicionales que tienen la forma siguiente:

operando1? operando2: operando3

La expresión operando1 debe ser una expresión booleana. La ejecución se realiza de la siguiente forma:

- Si el resultado de la evaluación de operando1 es verdadero, el resultado de la expresión condicional es operando2.

- Si el resultado de la evaluación de operando1 es falso, el resultado de la expresión condicional es operando3.

Ejemplo #3: Programa en C que muestra cuál es el mayor de dos números de tipo double.

```
//mayor.c
#include<stdio.h>
void main()
{
        double a = 10.2, b= 20.5, mayor =0;
        mayor = (a>b)? a : b;
        printf("Entre %g y %g el MAYOR es: %g\n",a,b,mayor);
}
```

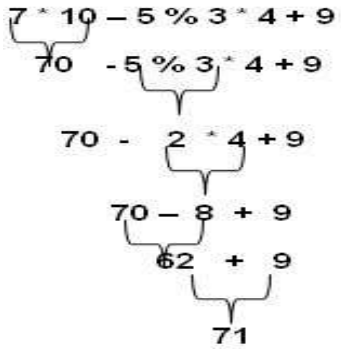
Ejemplo de Salida:

1.7.1 Prioridad y Orden de Evaluación en C:

La tabla que se presenta a continuación, resume las reglas de prioridad y asociatividad de todos los operadores. Las líneas se han colocado de mayor a menor prioridad. Los operadores escritos en la misma línea tienen la misma prioridad.

Operador	Asociatividad
() [] . ->	Izquierda a Derecha
! ++ -- sizeof	Izquierda a Derecha
* / %	Izquierda a Derecha
+ -	Izquierda a Derecha
< <= > >=	Izquierda a Derecha
== !=	Izquierda a Derecha
&&	Izquierda a Derecha
||	Izquierda a Derecha
?:	Derecha a Izquierda
= *= /= %= += -=	Derecha a Izquierda

Ejemplo: ¿Cuál es el resultado de la siguiente expresión: **7*10 – 5%3*4+9?**
Solución:



1.8 FUNCIONES DE ENTRADA Y SALIDA CON FORMATO:

Las funciones de entrada/salida permiten al programa comunicarse con el exterior. Son utilizadas para sacar determinadas informaciones por la pantalla y capturar valores por el teclado. Son estándar de la propia librería de C por lo que no hay necesidad de definirlas de nuevo.

1.8.1 Función printf( )

La función printf( ) permite escribir bytes en stdout (Monitor o Dispositivo de Salida estándar) usando formatos especificados. Esta función devuelve un entero igual al número de argumentos escritos. Se encuentra en el fichero de cabecera **stdio.h**.

Sintaxis
#include <stdio.h>
int printf (const char * formato[, argumentos]…);

Los formatos empleados son los siguientes:

Carácter	Salida
d	**(int)** entero con signo base 10
i	**(int)** entero con signo base 10
u	**(int)** entero sin signo base 10
o	**(int)** entero sin signo base 8
x	**(int)** entero sin signo base 16 (0,1,2 ..9, a,b,c…f)
X	**(int)** entero sin signo base 16 (0,1,2,…9, A,B,C …F)
f	**(double)** valor con signo, de la forma [-] dddd.dddd
e	**(double)** valor con signo, de la forma [-] d.dddde[±]ddd
E	**(double)** valor con signo, de la forma [-] d.ddddE[±]ddd
g	**(double)** valor con signo en formato f o e (el que sea más compacto para el valor y precisión dados)
G	**(double)** igual que g, pero introduce E en vez de e
c	**(int)** un solo carácter , correspondiente al byte menos significativo
s	cadena de carácter, escribe caracteres hasta encontrar el carácter nulo**0**

Un número colocado entre el % y el orden de formato, actúa como especificador de campo.

Ejemplo #4:

```
//impresión.c
#include<stdio.h>
void main()
{
        char nombre[25]={"Carlos"};
        int edad=25;
        float peso=55.4;
        printf("%s: Tiene %d anyos y pesa %f kilos\n",nombre,edad,peso);
}
```

Salida:

1.8.2 Función scanf( )

Para la entrada de los datos se usa la función scanf que lee byte (caracteres ASCII) de stdin (Teclado o Dispositivo de Entrada Estándar), los interpreta de acuerdo con el formato especificado y los almacena en los argumentos especificados.

Esta función devuelve un entero correspondiente al número de argumentos leídos y asignados, si el número devuelto es 0 indica que no puede leer ni asignar ningún dato; si se intenta leer un carácter de final de fichero (End of File) entonces retorna una constante EOF. Se encuentra en el fichero de cabecera **stdio.h**.

Sintaxis
#include<stdio.h>
int scanf (const char * formato[, argumento]…);

Los formatos empleados en la lectura de datos son los siguientes:

Carácter	El argumento es un puntero a	Entrada esperada
d	int	Entero con signo base 10.
o	int	Entero con signo base 8.
x, X	int	Entero con signo base 16.
i	int	Entero con signo base 10,8,16 . Si comienza con 0 es octal, y si comienza con 0x ,0X es hexadecimal.
U	Unsigned int	Entero sin signo base 10.
f,e,E,g y G	float	Valor con signo de la forma [-]d.dddd[{e|E}[+]ddd]
c	char	Un solo carácter.
s	char	Cadena de caracteres

Ejemplos de Entrada o Lectura de Datos:
scanf ("%d %f %c", &a,&b,&c); **// 5 23.4 +**
scanf ("%d, %f, %c", &a,&b,&c); **//5,23.4,+**
scanf ("%d: %f: %c", &a,&b,&c); **//5:23.4:+**

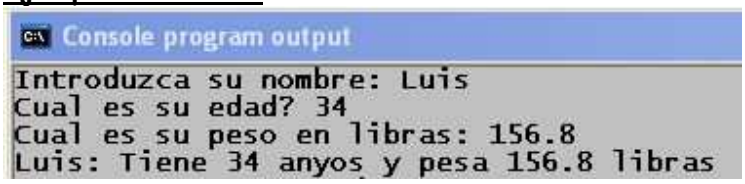
Es aconsejable poner espacios en blanco como separadores, si no tienen espacios en blanco al teclear se ponen estos espacios, para argumentos numéricos no hay problema pero para caracteres toma los espacios en blanco como parte del dato.

Ejemplo #5: Programa que realiza la lectura e impresión de datos.

//leer.c

```
#include<stdio.h>
void main()
{
        char nombre[25];
        int edad;
        float peso;
        printf("Introduzca su nombre: ");
        scanf("%s",nombre);
        printf("Cual es su edad? ");
        scanf("%d",&edad);
        printf("Cual es su peso: ");
        scanf("%f",&peso);
        printf("%s: Tiene %d anyos y pesa %f kilos\n",nombre,edad,peso);
}
```

Ejemplo de Salida:



Ejemplo #6: Programa en C que calcula el número de pulsaciones que una persona debe tener por cada 10 segundos de ejercicio, si la fórmula es:

num_pulsaciones = (220 - edad)/10

//pulsaciones.c

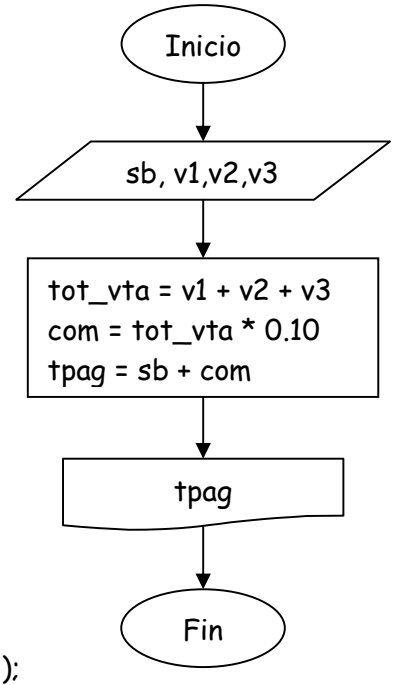
```
#include<stdio.h>
void main()
{
        int edad;
        float num_pulsaciones;

        printf("Introduzca su edad:");
        scanf("%d",&edad);
        num_pulsaciones = (220-edad)/10;
        printf("\nSus pulsaciones son: %f\n",num_pulsaciones);
}
```

EJERCICIOS RESUELTOS EN C:

1. Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuanto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

Solución en Pseudocódigo:

```
Inicio
        Leer sb, v1, v2, v3
        tot_vta = v1 + v2 + v3
        com = tot_vta * 0.10
        tpag = sb + com
        Imprimir tpag, com
Fin
```



//SueldoBase.c

```
#include<stdio.h>
void main()
{
        float sb,v1,v2,v3,tot_vta,com,tpag;
        printf("Introduzca el valor del Sueldo Base: ");
        scanf("%f",&sb);
        printf("Introduzca el valor de las tres ventas:  ");
        scanf("%f %f %f",&v1,&v2,&v3);
        tot_vta = v1 + v2 + v3;
        com = tot_vta * 0.10;
        tpag = sb + com;
        printf("El total a pagar es: %f\n",tpag);
        printf("El valor en comisiones es: %f\n",com);
}
```

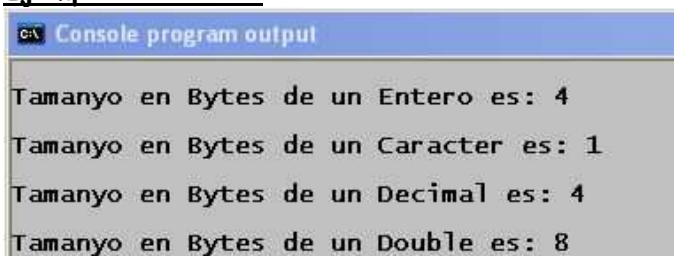
2. Mostrar en pantalla el tamaño en bytes de los tipos de datos existentes en C. Utilizar el operador sizeof() de la librería estándar.

//tamanyo_datos.c

```
#include<stdio.h>
void main( )
{
        printf("\nTamanyo en Bytes de un Entero es: %d\n",sizeof(int));
        printf("\nTamanyo en Bytes de un Caracter es: %d\n",sizeof(char));
        printf("\nTamanyo en Bytes de un Decimal es: %d\n",sizeof(float));
```

```
        printf("\nTamanyo en Bytes de un Double es: %d\n",sizeof(double));
}
```

Ejemplo de Salida:



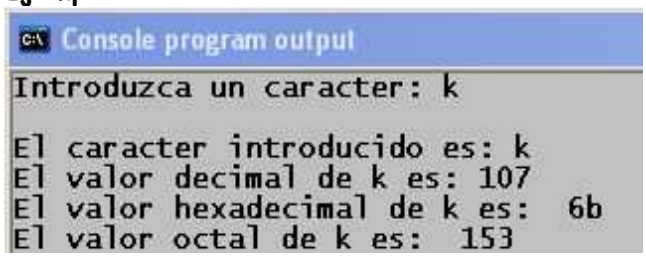
3. Introducir un carácter por teclado y mostrar en pantalla:

- ✓ El carácter introducido.
- ✓ El valor decimal del carácter introducido.
- ✓ El valor hexadecimal del carácter introducido.
- ✓ El valor octal del carácter introducido.

//conversión.c

```
#include<stdio.h>
void main()
{
        char caracter;
        printf("Introduzca un caracter: ");
        scanf("%c",&caracter);
        printf("\nEl caracter introducido es: %c",caracter);
        printf("\nEl valor decimal de %c es: %d",caracter,caracter);
        printf("\nEl valor hexadecimal de %c es:  %x",caracter,caracter);
        printf("\nEl valor octal de %c es:  %o\n",caracter,caracter);
}
```

Ejemplo de Salida:



4. Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuanto deberá pagar finalmente por su compra.

//Tienda.c

```
#include<stdio.h>
void main()
{
	float tc,d,tp;
	printf("Introduzca el total de la compra: ");
	scanf("%f",&tc);
	d = tc * 0.15;
	tp = tc - d;
	printf("El total a pagar es: %.2f",tp);
}
```

5. Un alumno desea saber cual será su calificación final en la materia de Algoritmos. Dicha calificación se compone de los siguientes porcentajes:

- 55% del promedio de sus tres calificaciones parciales.
- 30% de la calificación del examen final.
- 15% de la calificación de un trabajo final.

//Calificación.c

```
#include<stdio.h>
void main()
{
	int c1,c2,c3,ef,tf;
	float prom,ppar,pef,ptf,cf;

	printf("Introduzca el valor de las 3 calificaciones: ");
	scanf("%d %d %d", &c1,&c2,&c3);
	printf("Intrododuzca el valor del Examen Final: ");
	scanf("%d",&ef);
	printf("Intrododuzca el valor del Trabajo Final: ");
	scanf("%d",&tf);
	prom = (c1 + c2 + c3)/3;
	ppar = prom * 0.55;
	pef = ef * 0.30;
	ptf = tf * 0.15;
	cf = ppar + pef + ptf;
	printf("La calificacion final es: %.f",cf);
}
```

6. Un maestro desea saber que porcentaje de hombres y que porcentaje de mujeres hay en un grupo de estudiantes.

//Porcentaje.c

```
#include<stdio.h>
void main()
{
        int nh,nm,ta;
        float ph,pm;
        printf("Introduzca el numero de hombres : ");
        scanf("%d",&nh);
        printf("Introduzca el numero de mujeres : ");
        scanf("%d",&nm);
        ta= nh + nm;
        ph= nm *100 / ta;
        pm = nm *100 / ta;
        printf("El porcentaje de hombres que hay en el grupo de %d alumnos es %.f",ta,ph);
        printf("El porcentaje de mujeres que hay en el grupo de %d alumnos es %.f",ta,pm);
}
```

PROBLEMAS PROPUESTOS:

1. Elabore un programa que realice la conversión de kilogramos a libras sabiendo que 1 kg = 2.2046 libras. El usuario proporcionará el dato N kilogramos, el programa imprimirá el equivalente en libras.

2. El departamento de climatología ha realizado recientemente su conversión al sistema métrico. Diseñar un programa en C que realice las siguientes conversiones:

a. Leer la temperatura dada en la escala Celsius e imprimir en su equivalente Fahrenheit (la fórmula de conversión es "F=9/5 °C+32").

b. Leer la cantidad de agua en pulgadas e imprimir su equivalente en milímetros (25.5 mm = 1 pulgada).

3. Realice un programa en C que solicite al usuario el precio de un artículo sin IVA y calcule su valor aplicándole 15% de IVA. Imprimir el valor del IVA calculado y el nuevo precio del artículo.

4. Calcular el nuevo salario de un obrero si obtuvo un incremento del 25% sobre su salario anterior.

5. Tres personas deciden invertir su dinero para fundar una empresa. Cada una de ellas invierte una cantidad distinta. Obtener el porcentaje que cada quien invierte con respecto a la cantidad total invertida.

6. Diseñar un programa en C que lea dos valores enteros e imprima los resultados sumar, restar, dividir y multiplicar dicho número.

BIBLOGRAFÍA CONSULTADA:

- ✓ Asín Buñuel, José Luis. Lenguaje C. 2005.
- ✓ Ceballos, Francisco Javier: C/C++ Curso de Programación, 2da Edición. Editorial RAMA, 2002.
- ✓ Joyanes Aguilar, Luis; Zahonero Martínez Ignacio: Programación en C. McGraw Hill, 2001.
- ✓ Gottfried, Byron S: Programación en C. McGraw Hill, 1991.