**Capítulos:** Introducción, 1.1, 2.1-4.

## 1.1 Comencemos

Un programa en C consta de funciones y variables.

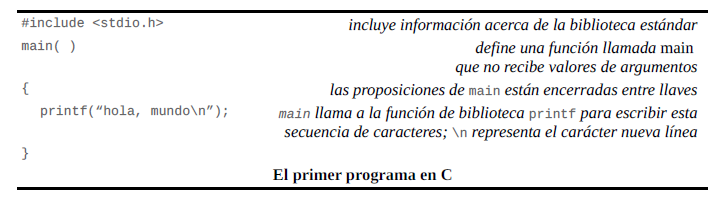
**Proposiciones:** Especifican las operaciones de cálculo que se van a realizar. Las funciones están formadas por proposiciones.

**Variables:** Almacenan los valores utilizados durante los cálculos.

**Función “main”:** Es una función especial, que es por donde inicia un programa. Esto significa que todo programa debe tener un main en algún sitio.

Por lo común, main llamará a otras funciones que ayuden a realizar su trabajo. Algunas de estas funciones serán escritas por el programador y otras serán de bibliotecas escritas previamente.

**Argumentos:** Es una lista de valores que sirven para comunicar datos entre las funciones. Se especifican entre paréntesis luego del nombre de la función.



## 2.1 Nombres de Variables

Los nombres se componen de letras y dígitos; el primer carácter debe ser una letra. El carácter de subrayado “\_” cuenta como una letra. Sin embargo, no se debe comenzar los nombres de variables con este carácter, puesto que las rutinas de biblioteca con frecuencia usan tales nombres. Las letras mayúsculas y minúsculas son distintas.

La práctica tradicional de C es usar letras minúsculas para nombres de variables, y todo en mayúsculas para constantes simbólicas.

## 2.2 Tipos y Tamaños de Datos

Tipos básicos de C

**char:** Un solo byte, capaz de contener un carácter del conjunto de caracteres locales.

**int:** Un entero, normalmente del tamaño natural de los enteros en la máquina en la que se ejecuta.

**float:** Punto flotante de precisión normal.

**double:** Punto flotante de doble precisión.

Además, existen algunos calificadores que se aplican a esos tipos básicos, **short** y **long** se aplican a enteros. La palabra int puede omitirse de tales declaraciones, lo que típicamente se hace.

A menudo **short** es de 16 bits y **long** de 32; **int** es de 16 o 32 bits. Cada compilador puede seleccionar libremente los tamaños apropiados para su propio hardware, siempre respetando: **short <= int <= long**.

El calificador **signed** o **unsigned** puede aplicarse a char o a cualquier entero. Los números **unsigned** son siempre positivos o cero y obedecen las leyes de la aritmética módulo 2n. Las variables **signed** obedecen al sistema complemento a dos.

El tipo **long double** especifica punto flotante de precisión extendida. Los archivos de encabezado headers estándar **<limits.h>** y **<float.h>** contienen constantes simbólicas para todos los tamaños.

## 2.3 Constantes

Una constante entera es un **int**.

Una constante **long** se escribe con una **l** o **L** terminal. Las constantes **sin signo** se escriben con una **u** o **U**, terminal. El sufijo **ul** o **UL** indica **unsigned long**.

Las constantes de **punto flotante** contienen un punto decimal o un exponente; su tipo es **double**. Los sufijos **f** o **F** indican una constante **float**; **l** o **L** indican un **long double.**

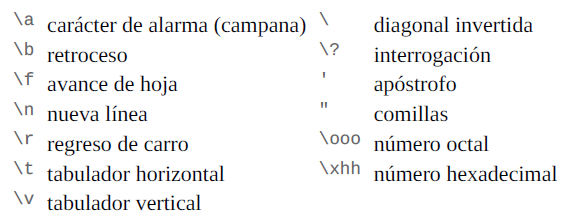
**Forma octal:** Un 0 (cero) al principio en una constante entera significa octal.

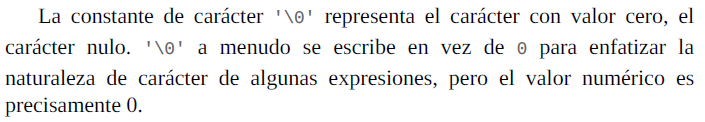
**Forma hexadecimal: 0x** o **0X** al principio significa hexadecimal.

Las constantes octales y hexadecimales también pueden ser seguidas por L y U.

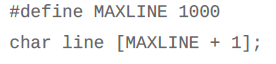
**Constante de carácter:** Es un entero, escrito como carácter dentro de apóstrofos, tal como ‘x’. El valor de una constante de carácter es el valor numérico del carácter en el conjunto de caracteres de la máquina. Por ejemplo, en el conjunto de caracteres ASCII el carácter constante ‘0’ tiene el valor 48, el cual no está relacionado con el valor numérico 0. Si escribimos ‘0’ en vez de un valor numérico que depende del conjunto de caracteres, el programa es independiente del valor particular y más fácil de leer. Normalmente se utilizan en comparaciones con otros caracteres.

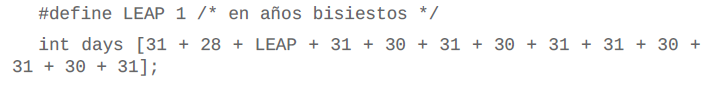
**Secuencias de escape:**



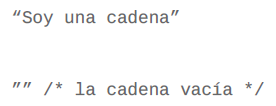


**Expresión constante:** Es una expresión que sólo inmiscuye constantes. Tales expresiones pueden ser evaluadas durante la compilación en vez de que se haga en tiempo de ejecución, y por tanto pueden ser utilizadas en cualquier lugar en que pueda encontrarse una constante.





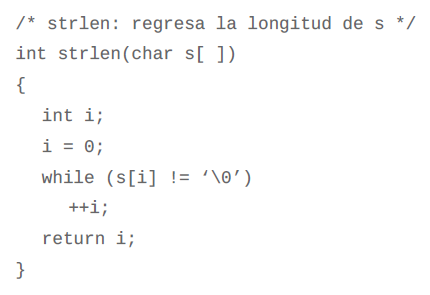
**Constante de cadena** o **Cadena literal:** Es una secuencia de cero o más caracteres encerrados entre comillas.



Las comillas no son parte de la cadena, sólo sirven para delimitarla. Las mismas secuencias de escape utilizadas en constantes de carácter se aplican en cadenas; **\”** representa el carácter comillas. Las constantes de cadena pueden ser concatenadas en tiempo de compilación.

Una constante de cadena es un arreglo de caracteres. La representación interna de una cadena tiene un carácter nulo **‘\0’** al final, de modo que el almacenamiento físico requerido es uno más del número de caracteres escritos entre las comillas. Por lo tanto, para determinar la longitud de una cadena, se debe leer completamente.

La función **strlen(s)** de la biblioteca estándar regresa la longitud del argumento s de tipo cadena de caracteres, excluyendo el ‘\0’ terminal.

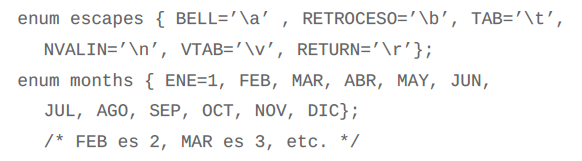


Se debe ser cuidadoso al distinguir entre una constante de carácter y una cadena que contiene un sólo carácter: **‘x’** no es lo mismo que **“x”**. El primero es un entero, utilizado para producir el valor numérico de la letra x en el conjunto de caracteres de la máquina. El último es un arreglo de caracteres que contiene un carácter y un ‘\0’.

**Constante de Enumeración:** Lista de valores enteros constantes.



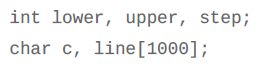
El primer nombre en un **enum** tiene valor 0, el siguiente 1, y así sucesivamente, a menos que sean especificados valores explícitos. Si no todos los valores son especificados, los valores no especificados continúan la progresión a partir del último valor que si lo fue.

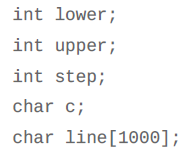


Los nombres que están en enumeraciones diferentes deben ser distintos.

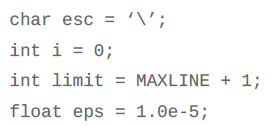
## 2.4 Declaraciones

Todas las variables deben ser declaradas antes de su uso. Una declaración especifica un tipo, y contiene una lista de una o más variables de ese tipo.





Una variable también puede ser inicializada en su declaración. Si el nombre es seguido por un signo de igual y una expresión, la expresión sirve como inicializador.



Si la variable no es automática, la inicialización es efectuada una sola vez, conceptualmente antes de que el programa inicie su ejecución, y el inicializador debe ser una expresión constante. Una variable automática explícitamente inicializada es inicializada cada vez que se entra a la función o bloque en que se encuentra.

Las variables estáticas y externas son inicializadas en cero por omisión. Las variables automáticas para las que no hay un inicializador explícito tienen valores indefinidos.

**const:** Calificador que puede aplicarse a la declaración de cualquier variable para especificar que su valor no será cambiado. Para un arreglo, el calificador const indica que los elementos no serán alterados.



La declaración const también se puede utilizar con argumentos de tipo arreglo, para indicar que la función no cambia ese arreglo.

