

República Bolivariana de Venezuela
Fundación Misión Sucre
Aldea Fray Pedro de Agreda
Introducción a la Programación III

Lenguaje C

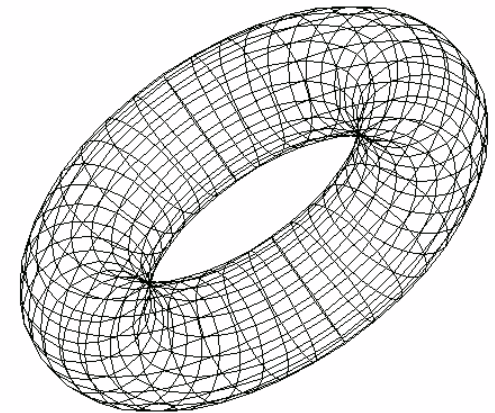
Puntos previos

- Los códigos fuentes generados en C requieren ser compilados para traducirlos a código de máquina.
- No confundir C con C++, no son lo mismo. Se puede decir que C++ es una derivación de C.
- Es un lenguaje sensible a mayúsculas (Case Sensitive). Cout es diferente a cout, Printf es diferente a printf
- Cada línea ejecutable debe finalizar con “;”.
- Lenguaje basado en funciones.
- Algunos programa escritos en C:
 - Núcleo de Linux

Estructura básica de un programa en C

La mejor forma de aprender un lenguaje es programando con él. El programa más sencillo que se puede escribir en C es el siguiente:

```
#include <stdio.h>
/*Esto es un comentario*/
int main( )
{
    printf("Misión Sucre");
    return 0;
}
```



Estructura básica de un programa en C

La sentencia *#include* `<stdio.h>`

El comando *#include*<> indica que un archivo será incluido al programa, en este caso `stdio.h` es un archivo o librería que contiene funcionalidades para realizar ciertas acciones. Por ejemplo mostrar mensajes por consola, captura de datos, etc.

Algunas librerías son:

- `stdio.h` : Funciones de entrada y salida.
- `stdlib.h` : Funciones estándares.
- `math.h` : Funciones matemáticas.
- `string.h` : Funciones de manejo de cadena de caracteres.

Ejercicio 1

- Según las librerías indicadas investigue al menos 3 funcionalidades de cada una de ellas:
 - `stdio.h`
 - `stdlib.h`
 - `math.h`
 - `string.h`

Estructura básica de un programa en C

int main() Es la función principal del programa. Todos los programas de C deben tener una función llamada *main()*. Es la que primero se ejecuta.

El *int* (viene de Integer=Entero) significa que cuando la función *main()* acabe devolverá un número entero. Este valor se suele usar para que un programa “padre” pueda saber cómo ha terminado el programa actual.

La función *main()* puede recibir a través de sus paréntesis parámetros de otros programas, sin embargo este tema será abordado en próximas secciones.

Estructura básica de un programa en C

Llaves: { } Son las llaves que indican el comienzo de una función, en este caso la función main().

Comentarios: son sentencias que no se ejecutan. Sirve para describir el programa.

Ejemplo:

```
/* Aquí un comentario del desarrollador */
```

Estructura básica de un programa en C

`return 0;` Es el valor de retorno de la función, y debe corresponder al tipo de retorno definido en la función `main`, `int main()`. Existe un tipo de retorno que permite no devolver valor, llamado `void` (vacío) y se utiliza `void main()`.

Mas adelante se nombrarán los tipos de datos utilizados en C.

Estructura básica de un programa en C (Cont.)

El nombre de una función en C siempre va seguida de paréntesis, para la recepción de parámetros. La definición de la función está formada por un bloque de sentencias, que esta encerrado entre llaves {}.

```
#include <stdio.h>
```

```
int main( )
```

```
{
```

```
    printf("Saludos a la Misión Sucre");
```

```
    return 0;
```

```
}
```



Función printf()

printf() es una función definida dentro de librería stdio.h la cual permite mostrar mensajes por la consola. Su sintaxis básica es:

```
printf( "Mensaje a mostrar por consola" );
```

Ejercicio 2

Ubique la librería stdio.h y revise las diferentes sintaxis de la función printf();

Compilación en C

```
gcc -c primero.c -o primero.exe
```

Tipos básicos y variables

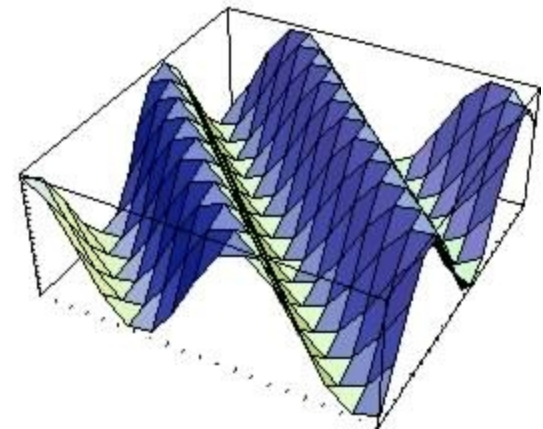
Los tipos de datos básicos definidos por C son caracteres, números enteros y números en coma flotante. Los caracteres son representados por char, los enteros por short, int, long y los números en coma flotante por float y double. Los tipos básicos disponibles y su tamaño son:

Char	Carácter	(normalmente 8 bits)
Short	Entero corto con signo	(normalmente 16 bits)
Int	Entero con signo	(depende de la implementación)
Unsigned	Entero sin signo	(depende de la implementación)
Long	Entero largo con signo	(normalmente 32 bits)
Float	Flotante simple	(normalmente 32 bits)
Double	Flotante doble	(normalmente 64 bits)

Tipos básicos y variables

Las variables son definidas utilizando un identificador de tipo seguido del nombre de la variable. Veamos el siguiente programa:

```
#include <stdio.h>  
main()  
{  
  
    float cels, farh;  
  
    farh = 35.0;  
  
    cels = 5.0 * ( farh - 32.0 ) / 9.0;  
  
    printf("-> %f F son %f C\n", farh, cels );  
  
}
```



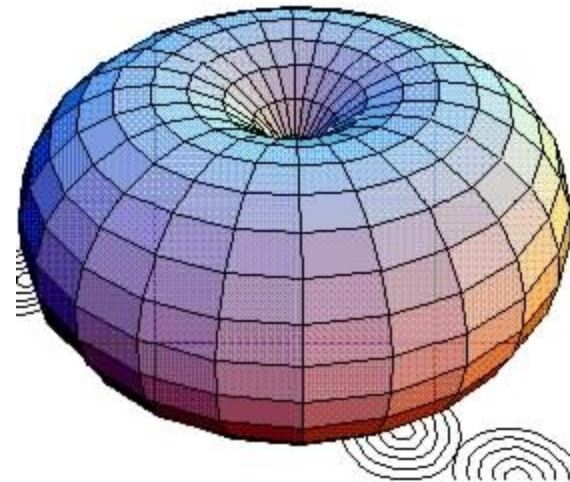
Expresiones y operadores

Los distintos operadores permiten formar expresiones tanto aritméticas como lógicas. Los operadores aritméticos y lógicos son:

+, -	suma, resta
++, --	incremento, decremento
*, /, %	multiplicación, división, módulo
>>, <<	rotación de bits a la derecha, izquierda.
&	AND booleano
	OR booleano
^	EXOR booleano
~	complemento a 1
!	complemento a 2, NOT lógico
==, !=	igualdad, desigualdad
&&,	AND, OR lógico
<, <=	menor, menor o igual
>, >=	mayor, mayor o igual

Expresiones y operadores

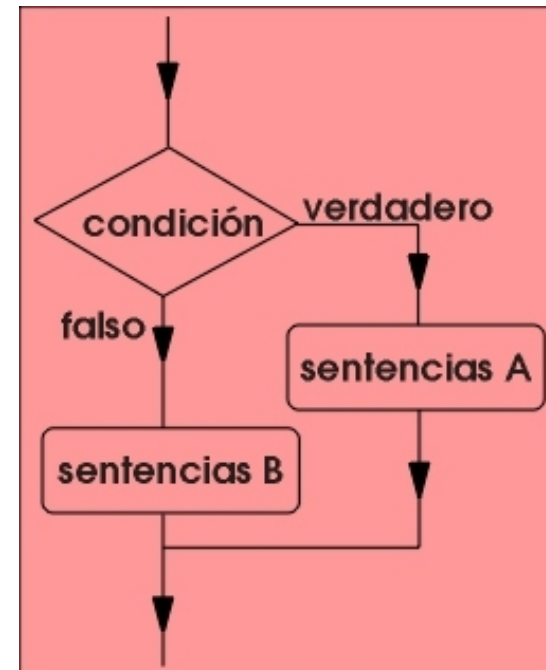
En la evaluación de expresiones lógicas, los compiladores normalmente utilizan técnicas de evaluación rápida. Para decidir si una expresión lógica es cierta o falsa muchas veces no es necesario evaluarla completamente. Por ejemplo una expresión formada $\langle \text{exp1} \rangle \parallel \langle \text{exp2} \rangle$, el compilador evalúa primero $\langle \text{exp1} \rangle$ y si es cierta, no evalúa $\langle \text{exp2} \rangle$.



Sentencia if

La sentencia de control básica es `if (<e>) then <s> else <t>`. En ella se evalúa una expresión condicional y si se cumple, se ejecuta la sentencia `s`; si no, se ejecuta la sentencia `t`. La segunda parte de la condición, `else <t>`, es opcional.

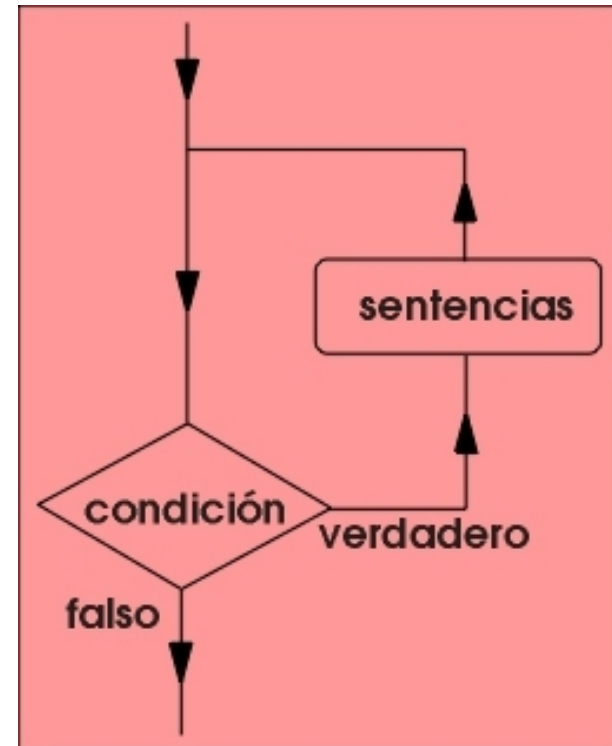
```
if ( condición )  
{  
    Instrucciones  
}  
else  
{  
    Otras Instrucciones  
}
```



Sentencia while

```
Int tamano = 100;
```

```
while ( r < tamano )  
{  
    r++;  
}
```



Sentencia For

```
for (int i=0; i< 10; i++)  
{  
    Sentencias  
}
```

