503208 Programación I 2021-I

Laboratorio 3 — 09 de Abril, 2021

Aníbal Tolosa

Temas

- Operadores:
  - Operadores lógicos.
- Estructuras de control:
  - Condicionales:
    - `if`
    - `else`
    - `else if`
  - Bucles:
    - `while`
    - `do while`
    - `for`

1. Operadores

1.1. Operadores Lógicos en C

Los operadores lógicos son operadores que evalúan a un valor de verdad VERDADERO o FALSO (valores *booleanos*).

Operación	**Operador**	**Ejemplo**	**Resultado**
Comparación: Igual	`==`	`a==b`	$\neq 0$ si $a = b$, 0 caso contrario.
Comparación: No igual	`!=`	`a!=b`	$\neq 0$ si $a \neq b$, 0 caso contrario.
Comparación: Mayor	`>`	`a>b`	$\neq 0$ si $a > b$, 0 caso contrario.
Comparación: Mayor o igual	`>=`	`a>=b`	$\neq 0$ si $a \geq b$, 0 caso contrario.
Comparación: Menor	`>`	`a<b`	$\neq 0$ si $a < b$, 0 caso contrario.
Comparación: Menor o igual	`<=`	`a<=b`	$\neq 0$ si $a \leq b$, 0 caso contrario.
Negación	!	`!a`	0 si $a \neq 0$, $\neq 0$ caso contrario.
Y lógico	`&&`	`a && b`	$\neq 0$ si $a \neq 0$ y $b \neq 0$, 0 en otro caso.
O lógico	`||`	`a || b`	$\neq 0$ si $a \neq 0$ y/o $b \neq 0$, 0 si $a = b = 0$.

* *En **C**, FALSO es representado con un valor 0, VERDADERO es cualquier valor $\neq 0$.*

Dado que el estándar sólo asegura el valor de FALSO, es buena práctica comparar los valores de verdad con $= 0$ o $\neq 0$.

* *Notar que en **C** no tenemos un operador **O exclusivo** lógico.*

2. Estructuras de control

Las estructuras de control son declaraciones dentro de un programa que permiten modificar el orden de ejecución de éste.

2.1. Condicionales

2.1.1. `if`

La declaración `if` permite controlar si ejecutar o no una sección de código según una condición. La condición es alguna expresión que debe evaluar a FALSO (lo que es representado por el valor 0) o VERDADERO (que es representado por cualquier valor $\neq 0$).

if (*condición*) {
  `/* codigo */`
}

Por ejemplo:

```
if (1) {
    printf("Este texto se mostrara en consola.\n");
}
```

2.1.2. `else if`

La declaración `else if` nos permite condicionar la ejecución de un segundo bloque de código, sujeto a un `if`. Si la condición en el `if` no se cumple, se evalúa la condición en el `else if`, ejecutándose su bloque de código si la condición es VERDADERA. Pueden ir múltiples `else if` seguidos, donde se ejecuta el primero que se cumpla.

if (*condición*) {
  `/* codigo 1 */`
}
else if (*condición 2*) {
  `/* codigo 2 */`
}

Por ejemplo:

```
if (0) {
    printf("Este texto NO se mostrara en consola.\n");
}
else if (!1) {
    printf("Este tampoco.\n");
}
else if (!0) {
    printf("Este texto SI aparecera.\n");
}
```

2.1.3. `else`

Para los casos donde exista un `if`, la declaración else permite ejecutar código en el caso donde la condición expresada en el `if` o `if else` anteriores no se hayan cumplido. Vice versa, si se ejecutó algún bloque anterior correspondiente, el código de la declaración `else` no se ejecutará.

if (*condición*) {
  codigo 1
}
else {
  codigo 2
}

Por ejemplo:

```
if (0) {
    printf("Este texto NO se mostrara en consola.\n");
}
else if (1) {
    printf("Este texto SI aparecera.\n");
}
else {
    printf("Este texto NO aparecera.\n");
}
```

2.2. Bucles

Los bucles son partes del programa que se pueden ejecutar múltiples veces consecutivas.

2.2.1. `while`

La declaración `while` ejecuta una sección de código mientras que su condición lógica especificada sea VERDADERA. Una vez la condición no se cumpla más —la condición evalúe a FALSO—, se reanuda la ejecución del resto del programa.

while (*condición*) {
  / codigo */*
}

Por ejemplo:

```
    i = 1;
    while (i <= 3) {
        printf("Este printf() se ejecutara 3 veces. %d\.n",i);
        i++;
    }
```

Despliega:

```
Este printf() se ejecutara 3 veces. 1
Este printf() se ejecutara 3 veces. 2
Este printf() se ejecutara 3 veces. 3
```

2.2.2. `do while`

En el caso del **`while`**, la condición se evalúa al inicio de la ejecución del bucle. Con **`do while`**, la condición se evalúa luego de la ejecución del bucle.

do {
  / codigo */*
} **while** (*condición*);

Por lo tanto, se asegura —por lo menos— una iteración del bucle.

Por ejemplo:

```
do {
    printf("Este bucle ejecutara una iteracion.\n");
} while (0);
```

2.2.3. `for`

El bucle for tiene tres partes:

for (*inicialización*; *condición*; *actualización*)

- La INICIALIZACIÓN se ejecuta al iniciar el bucle. Configura las condiciones iniciales de éste.
- La CONDICIÓN se comprueba al iniciar el bucle (luego de la inicialización), y después de cada actualización.
- La ACTUALIZACIÓN se ejecuta al finalizar cada iteración del bucle.

for (*inicialización*; *condición*; *actualización*){
  / codigo 1 */*
}

Por ejemplo, el bucle mostrado anteriormente con **while**, ahora escrito con **for**:

```
for (i = 1; i<=3; i++)
    printf("Este printf() se ejecutara 3 veces. %d\.n",i);
```

El resultado es el mismo.

Ejercicios

1. Crear un programa que reciba un caracter por consola, y responda si el caracter es número, letra mayúscula, letra minúscula u otro símbolo.
2. Crear un programa que responda, sobre tres números a, b, y c leídos por consola, cual de ellos es el mayor.
3. Crear un programa que reciba dos números enteros x e y, y un caracter *op*, describiendo una operación matemática básica, en el siguiente formato:

   $$x\ op\ y$$

   op puede ser uno de los siguientes caracteres: +, -, *, /.
   Luego, desplegar en consola el resultado de la operación descrita por la entrada. En el caso de la división, desplegar el cuociente y el resto.
4. Crea un programa que reciba un número entero n por consola, luego despliegue los numeros desde el 1 hasta n.
5. Crea un programa que, dados dos enteros n y m por consola, $n \leq m$, despliege los numeros desde m hasta n.
6. Crea un programa que reciba un entero, calcule y despliegue en pantalla su factorial.
7. Crea un programa que, dado un entero n, $n \geq 2$, dibuje un cuadrado hueco de tamaño n en pantalla. Por ejemplo, para $n = 6$:

```
* * * * * *
*         *
*         *
*         *
*         *
* * * * * *
```

8. Crea un programa que, dado un entero n, $n \geq 2$, dibuje un triángulo rectángulo de altura n en pantalla. Por ejemplo, para $n = 5$:

```
        *
      * *
    * * *
  * * * *
* * * * *
```

9. Crea un programa que dado un entero n por consola, calcule y despliegue en pantalla si éste es un número primo o no.
10. Crea un programa que muestre una tabla de caracteres (*0-9, A-Z, a-z*), con sus valores en ASCII en decimal y hexadecimal