INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

upiita-ipn

# Unidad 1: Introducción a la programación.


Nombre: Arturo Matamoros Balderas

2019640534

Grupo: 1MV1


Materia: Introducción a la programación

## Introducción

El lenguaje C facilita un método estructurado y disciplinado para el diseño de programas. En esta Unidad aplicaremos la programación en C y se realizarán ejercicios los cuales representarán características importantes de C.

Talvez este sea el punto más importante, debido a que es el primer paso que se está dando hacia la programación en lenguaje C, se requiere tener una noción clara de la estructura fundamental y ejercicios básicos para el desarrollo de las siguientes unidades con los ejercicios solicitados en cada una de ellas.

Explicaremos la importancia de la programación estructurada para producir programas que sean claros, corregibles, que se pueden mantener y que probablemente funcionen al primer intento.

## Objetivo

- Escribir programas sencillos en C.
- Utilizar instrucciones sencillas de entrada y salida.
- Utilizar los operadores aritméticos.
- Diseñar y codificar un programa que englobe lo visto en la unidad_1 comprendiendo el funcionamiento del mismo.

## Desarrollo

### Estructura de un Programa en C

Se pueden observar diferentes tipos de lenguajes de programación nombrados de *alto nivel,* con estos nos referimos a aquellos programas que están lejanos al lenguaje maquina o ensamblador que es el que utiliza el procesador para ejecutar todos los programas.

En un programa en C, se incluye la función main se incluye una primera sentencia que llama a la función printf. Esta toma como argumento (encerrado entre paréntesis) una cadena de caracteres limitados por dobles comillas (“ ”) y la imprime en la salida habitual, que generalmente es el terminal en el que trabajamos. El símbolo \n indica un cambio de línea.

La segunda sentencia, return 0, termina el programa y devuelve un valor (cero) al sistema operativo (por lo general es cero si la ejecución fue correcta y usan valores distintos de cero para indicar diversos errores que pudieran ocurrir). Si bien no es obligatorio terminar el programa con un programa, si la finalización ha tenido éxito o no. De cualquier manera, en este caso, si sacamos esa sentencia el programa, este funcionará exactamente igual, pero al ser compilado, el compilador nos advertirá de la falta de retorno.

Cuando incluimos comentarios en un programa es una saludable práctica, como lo reconocerá cualquiera que haya tratado de leer un listado hecho por otro programador o por si mismo, varios meses atrás. Para el compilador, los comentarios son inexistentes, por lo que no generan líneas de código, permitiendo

abundar en ellos tanto como se desee. Esto se delimita por /* El comentario que se dese colocar */

Es importante que cada sentencia que se escriba en el lenguaje C sea finalizada por (;) esto indica el cierre de la instrucción que se está escribiendo.

## Uso de variables y constantes

Las variables son altamente imprescindibles al momento de programar, de hecho, sería imposible conseguir una aplicación con una funcionalidad clásica sin usar variables; por esta misma razón es necesario aprender a usarlas bien y lo tenemos muy fácil, pues su uso es bastante sencillo e intuitivo, tanto para declararlas como para asignarles valores

Las variables son posiciones en memoria donde estarán guardados los diferentes valores que le damos o que toman durante la ejecución los datos que usamos y normalmente estarán disponibles a lo largo de la ejecución de nuestro programa. Para asignar valores a una variable en una gran variedad de lenguajes que incluye a C++ se usa el operador “=” seguido del valor que le daremos a la variable (no necesariamente se tiene que usar “=” para realizar esta acción).

Los datos constantes, o también coloquialmente conocido como “variables constantes” tienen un valor fijo durante toda la ejecución del programa, es decir, este valor no cambia ni puede ser cambiado a lo largo de la ejecución de nuestro programa. Las constantes son muy útiles para especificar el tamaño de un vector y para algunas otras cosas, como facilidad de uso y confiabilidad del código. Para declarar una constante, se hace después de declarar las librerías y antes de las funciones, la sintaxis es la siguiente:

```
#define nombre _constante valor.
```

La instrucción #define nos permite declarar constantes de una manera rápida y sencilla. Hay que tener en cuenta que al declarar constantes con #define debemos hacerlo después de los #include para importar librerías, pero antes de declarar nuestras funciones y demás.

## Creación de código fuente, ejecutable y objeto

**Código fuente:** Conjunto de sentencias entendibles por el programador que componen el programa o una parte de ello. Suele estar almacenado en un fichero del tipo texto como los que se pueden abrir, por ejemplo, con el bloc de notas o Wordpad en los entornos Windows. El código fuente estará en un lenguaje de programación determinado, elegido por el programador, como puede ser: Basic, C,C++;C# Java, Perl ,Phython, PHP.

**Código objeto**: Conjunto de instrucciones y datos escritos en un lenguaje que entiende el ordenador directamente: binario o código máquina. Provienen de la traducción de cierto código fuente, es un fragmento del programa final y es especifico de la plataforma de ejecución.

**Código ejecutable:** Reúne diferentes códigos u objetos generados por los programadores junto con las “librerías de uso general” (propia del entorno o del lenguaje de programación) componiendo el programa final. Este es el código que ejecutan los usuarios del sistema, y es especifico para una plataforma concreta: Windows, Linux, Mac OS.

# Operadores

## ¿Qué es un operador?

Un operador es un elemento de programa que se aplica a uno o varios operadores en una expresión o instrucción. Los operadores que requieren un operando, como el operador de incremento se conocen como operadores unarios. Los operadores que requieren dos operandos, como es el caso de los operadores aritméticos (+,-,*,/) se conocen como operadores binarios. Un operador, el operador condicional (?:), utiliza tres operandos y es el único operador ternario de C++.

## Tipos de operadores

Existen tipos de operadores según su función, que son aritméticos, relacionales, de asignación, lógicos de dirección y de manejo de Bits.

| Operador | Acción | Ejemplo | Resultado |
|---|---|---|---|
| = | Asignación Básica | X = 6 | X vale 6 |
| *= | Asigna Producto | X *= 5 | X vale 30 |
| /= | Asigna División | X /= 2 | X vale 3 |
| += | Asigna Suma | X += 4 | X vale 10 |
| -= | Asigna Resta | X -= 1 | X vale 5 |
| %= | Asigna Modulo | X %= 5 | X vale 1 |
| <<= | Asigna Desplazamiento Izquierda | X <<= 1 | X vale 12 |
| >>= | Asigna Desplazamiento Derecha | X >>= 1 | X vale 3 |
| &= | Asigna AND entre Bits | X &= 1 | X vale 0 |
| ^= | Asigna XOR entre Bits | X ^= 1 | X vale 7 |
| \|= | Asigna OR entre Bits | X \|= 1 | X vale 7 |

| Operador | Acción | Ejemplo | Resultado |
|---|---|---|---|
| - | Resta | X = 5 - 3 | X vale 2 |
| + | Suma | X = 5 + 3 | X vale 8 |
| * | Multiplicación | X = 2 * 3 | X vale 6 |
| / | División | X = 6 / 3 | X vale 2 |
| % | Módulo | X = 5 % 2 | X vale 1 |
| -- | Decremento | X = 1; X-- | X vale 0 |
| ++ | Incremento | X = 1; X++ | X vale 2 |

| Operador | Acción | Ejemplo |
|---|---|---|
| - | Desplazamiento descendente | pt1 - n |
| + | Desplazamiento ascendente | pt1 + n |
| - | Distancia entre elementos | pt1 - pt2 |
| -- | Desplazamiento descendente de 1 elemento | pt1-- |
| ++ | Desplazamiento ascendente de 1 elemento | pt1++ |

| Operador | Relación | Ejemplo | Resultado |
|---|---|---|---|
| < | Menor | X = 5; Y = 3;<br>if(x < y) x+1; | X vale 5<br>Y vale 3 |
| > | Mayor | X = 5; Y = 3;<br>if(x > y) x+1; | X vale 6<br>Y vale 3 |
| <= | Menor o igual | X = 2; Y = 3;<br>if(x <= y) x+1; | X vale 3<br>Y vale 3 |
| >= | Mayor o igual | X = 5; Y = 3;<br>if(x >= y) x+1; | X vale 6<br>Y vale 3 |
| == | Igual | X = 5; Y = 5;<br>if(x == y) x+1; | X vale 6<br>Y vale 5 |
| ¡= | Diferente | X = 5; Y = 3;<br>if(x ¡= y) y+1; | X vale 5<br>Y vale 4 |

| Operador | Acción | Ejemplo | Resultado |
|---|---|---|---|
| && | AND Lógico | A && B | Si ambos son verdaderos se obtiene verdadero(true) |
| \|\| | OR Lógico | A \|\| B | Verdadero si alguno es verdader |
| ! | Negación Lógica | !A | Negación de a |

| Operador | Acción | Ejemplo | Resultado |
|---|---|---|---|
| << | Desplazamiento a Izquierda | a << b | X vale 2 |
| >> | Desplazamiento a Derecha | X = 5 + 3 | X vale 8 |
| ~ | Complemento | X = 2 * 3 | X vale 6 |
| & | AND | X = 2 & -2 | X vale 2 |
| ^ | XOR | X = 7 ^ -2 | X vale -7 |
| \| | OR | X = 6 \| 13 | X vale 15 |

| Operador | Acción | Ejemplo | Resultado |
|---|---|---|---|
| << | Desplazamiento a Izquierda | a << b | X vale 2 |
| >> | Desplazamiento a Derecha | X = 5 + 3 | X vale 8 |
| ~ | Complemento | X = 2 * 3 | X vale 6 |
| & | AND | X = 2 & -2 | X vale 2 |
| ^ | XOR | X = 7 ^ -2 | X vale -7 |
| \| | OR | X = 6 \| 13 | X vale 15 |

### Expresiones simples y complejas

Una expresión es una secuencia de uno o mas operadores y cero o mas operadores que se pueden evaluar como un valor, objeto método o espacio de nombres único. Las expresiones pueden constar de un valor literal, una invocación de método, un operador y sus operandos o un nombre simple. Los nombres simples pueden ser el nombre de una variable, el miembro de un tipo, el parámetro de un método, un espacio de nombre o un tipo.

Las expresiones pueden usar operadores que, a su vez usan otras expresiones como parámetros o llamadas a métodos cuyos parámetros son, a su vez, otras llamadas a métodos, de modo que pueden variar de simples a muy complejas.

## Conclusiones

Es importante comprender la estructura básica de in programa en c debido a que su forma nos ayudara a comprender más adelante temas más complejos abarcando desde funciones, cadenas, apuntadores, puertos.

Los operadores se ocuparán siempre en cualquier programa en C o C++ esto debido a que representan las operaciones mas simples a su ves que las operaciones mas complejas se componen por los operadores.

La importancia de las variables es muy grande y de mucho peso debido con estas variables se trabajara dentro del programa en C o C++, ya que para evaluar o determinar el comportamiento de un sistema se tiene que guardar en una variable, dependiendo del uso posterior que se le va dar: si el caso es que se pretende utilizar en todos los cálculos fundamentales la mejor opción es utilizar una constante, en dado caso que no se utiliza una variable ya sea de estilo: entero, flotante, doble o carácter.

## Bibliografía

Guía para el programador para el IBM PC y PS/2. Peter Norton, Richard Wilton. Anaya Multimedia.

Programación I. José A. Cerrada, Manuel Collado. UNED

Paul J. Deitel. (2004). Introduccion a la programacion en C. En C/C++ como programar (23-35). Mexico: Pearson Education.

http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/control/lengua_C/programa.htm

https://www.programarya.com/Cursos/C++/Sistema-de-Tipos/Variables-y-Constantes

http://profesores.fi-b.unam.mx/carlos/lcpi/p09/OPERADORES%20EN%20%20C++.pdf