Autoestudio 2: Variables y Operadores

## Componentes de un programa

Cuando escribimos en un lenguaje de programación tenemos que entender la sintaxis del lenguaje.

A continuación empezaremos con la sintaxis básica de **C**, primero presentando elementos separados y después lo iremos juntando.

Pongamos que tenemos el siguiente código:

**int** num **=** 100**;**

* ***Tipo de Dato***: indica con qué tipo de información estaremos trabajando, es decir si son números enteros o decimales, si son letras u otro tipo.

Ejemplos: int, char, float, etc…

En la siguiente sección veremos a detalle los principales tipos de datos de maneja el lenguaje C.

* ***Identificador:*** Son nombres dados a constantes, variables, tipos y funciones de un programa. Se forman con la combinación de letras, números y el símbolo ‘\_’. El primer carácter debe ser una letra.

Ejemplos: pi, vocales, x, i, num, miIdentificador, mi\_identificador, num1, num\_2, etc…

* ***Constante***: Una constante es un valor que no cambia, por ejemplo esto puede ser un valor, como un número o una letra.

Ejemplos: 3.1416, ‘a’, 44, “perro”, etc …

* ***Operador:***  Son al igual que en las matemáticas símbolos que nos indican que una operación ocurre, hay operadores de varios tipos: asignación, comparación, aritméticos y lógicos.

Ejemplos: -, \*, +, =, <, >, /, %, etc…

Más adelante en el autoestudio veremos a detalle los principales operadores del lenguaje C.

* ***Instrucción***: Unidad ejecutable más pequeña de un programa. Las instrucciones controlan el flujo u orden de ejecución, la unidad básica para separar instrucciones es ‘***;***’

Ejemplos: int num = 100; char letra = ‘a’; printf(“hola”); scanf(“%i%\*c”, &letras); etc....

* ***Variable:*** Es un recipiente de memoria que puede guardar un valor de forma temporal, la variable se compone de un tipo de dato nombrada con un identificador.

Ejemplos: int num, float decimal, int a, int me\_gusta, char no\_me\_gusta, char letra, etc…

Un programa en sí se compone de muchas instrucciones que a su vez son operaciones con variables, el aprender a manejar esto es simple, siempre y cuando se cuente con una idea clara de lo que se desea hacer. Esta idea clara y específica es el algoritmo.

## Declaración de Variables:

Antes de hacer operaciones con las variables, necesitamos decirle al programa qué variables existen y qué tipo de información representan, a este proceso le llamamos declarar variables. **Declarar una variable** significa: a) **definir un tipo,** b) darle un nombre o **identificador** a la variable y opcionalmente c) **su valor inicial**. Las formas para declarar una variable son las siguientes:

* **tipo\_básico identificador**

*Ejemplos*:

**int** entero**;**

**char** vocal;

**float** real;

* **tipo\_básico identificador [ = valor];**

*Ejemplos*:

**int** entero =10**;**

**char** vocal = ‘a’;

**float** real = 3.14159;

Más de una variable puede ser declarada en la misma instrucción:

* **tipo\_de\_dato nombre\_1[ , nombre\_2, nombre\_3, …, nombre\_n]**

*Ejemplos*:

**int** entero, contador, dia**;**

**int** dia=27, contador=0;

**char** vocal=‘a’, vocal2= ‘e’, consonante = ‘b’;

**int** num1, num2 = 0, num3;

**float** pi = 3.14159, numReal1 = 1.0, num\_real2;

Los 3 tipos de datos principales que manejaremos en el curso son int, float, y char, aunque cabe mencionar que éstos no son los únicos tipos de dato que existen.

### int

El tipo de dato **int** permite manejar números enteros, positivos y negativos. *Ejemplos*:

**int** num1 = 0;

**int** num2 = 34;

**int** num3 = -10;

**int** num4 = 1;

**int** num5 = 100;

**int** num6 = -200;

### float

El tipo de dato **float** permite manejar números reales, positivos y negativos. *Ejemplos*:

**float** numReal1 = 0.0;

**float** numReal2 = 34.3;

**float** numReal3 = -10.5001;

**float** numReal4 = 1.10;

**float** numReal5 = 100.034;

**float** numReal6 = -200.999;

### char

El tipo de dato **char** permite manejar caracteres (letras), para ello se deben escribir entre comillas simples (‘a’, ‘b’, ‘c’, etc.)

En el fondo el tipo char se trata de un tipo numérico. Los valores de tipo carácter se pueden utilizar como números, ya que para la computadora todos los caracteres tienen un número asignado.

**char** caracter;

**int** entero;

caracter = 65; // valdría como una ‘A’

entero = ‘A’; // valdría como un 65

## Inicializar variable

Cuando se le asigna el valor a una variable se dice que **se inicializa**, si a una variable no se le ha asignado un valor se dice que ésta **no ha sido inicializada**.

|  |  |
| --- | --- |
| Variables sin Inicializar:  int a;  Char b;  float c; | Variables Inicializadas:  int a =10;  Char b = ‘a’;  float c = 1.23; |

Ejemplo:

# Variables Constantes y Dinámicas

Hay dos tipos de variables, las **variables dinámicas** como las que acabamos de presentar y las **variables constantes**, la diferencia radica en que una variable dinámica puede cambiar su valor después de ser declarada y la constante no. Una variable constante se declara con el prefijo **const**

*Ejemplos*:

**const int** entero =10**;**

Una vez declarada la variable constante ‘entero’ ya no podrá ser modificada en el resto del programa.

## Tipos de Operadores

### Aritméticos

**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.** Si ambos operadores son enteros el resultado es entero. En el resto de los casos el resultado es real.

**%** **Módulo o resto (residuo) de una división.** Los operandos tienen que ser enteros

### Lógicos o Booleanos

**Operador Operación**

**&&** **AND.** Da como resultado el valor lógico1 si ambos operandos son distintos de

cero.

**||** **OR.** Da como resultado el valor lógico 0 si ambos operandos son cero.

**!** **NOT.** El resultado es 0 si el operando tiene un valor distinto de cero, y en

caso contrario.

### Comparación

**Operador Operación**

**<** Primer operando **menor que** el segundo

**>** Primer operando **mayor que** el segundo

**<=** Primer operando **menor o igual que** el segundo

**>=** Primer operando **mayor o igual que** el segundo

**==** Primer operando **igual que** el segundo

**!=** Primer operando **distinto que** el segundo

### Asignación

**Operador Operación**

**++** Incremento y asignación.

**--** Decremento y asignación.

**=** Asignación simple.

**\* =** Multiplicación más asignación

**/=**  División más asignación

**+=** Suma más asignación

**- =** Resta más asignación

Ejemplo de instrucciones operadores aritméticos:

int num1 , num2, num3, num4, num5;

num1 = 100 +8 - 5;

num2 = 100 - 100;

num3 = 100 \* 10 / 51;

num4 = 100 % 10 + 19;

num5 = num1 +num2 +num3 \*num4;

¿Qué valor guarda cada variable al final?

# Escribiendo y leyendo de Consola

Para escribir mensajes y datos en la pantalla de texto se utiliza la primitiva **printf**

**printf** (“Hola!!!!”) ;

La primitiva printf también puede escribir el valor de variables en la pantalla de texto.Para poder imprimir el valor de una variable en la pantalla de texto es necesario indicar el formato de los datos a escribir.   
.

|  |  |
| --- | --- |
| **Tipo de datos** | **Formato entrada/salida** |
| char (carácter) | %c |
| int (entero) | %i o %d |
| float (punto flotante o decimal) | %f |

.

Al final del mensaje que imprime el valor de la variable hay que indicar la lista de variables a imprimir. La lista de variables deberá coincidir con el tipo de datos que se desea imprimir.

Supongamos que **var\_a** es una variable de tipo entero con un valor de 5 y **var\_b** una variable de tipo flotante con un valor de 23.9. El siguiente mensaje imprime en la pantalla de texto el valor de las variables **var\_a** y **var\_b**.

**printf** (“El valor de var\_a = %i y el de var\_b = %f”, var\_a, var\_b);

La siguiente instrucción es incorrecta. ¿Porqué?

**printf** (“El valor de var\_a= %i y el de var\_b = %f”, var\_b, var\_a);

El lenguaje de programación C proporciona una primitiva para leer valores desde el teclado. Los valores que puede leer están asociados con los tipos de datos (char, int, float, etc.). Para leer un valor es necesario indicar el formato del tipo básico de datos. La primitiva para leer es : **scanf**

Todo scanf deberá estar asociado con un printf previo. El printf se utiliza para informarle al usuario lo que debe insertar. El formato general del scanf es :

**scanf** (“%*Formato\_Tipo*\*c”,  &Variable);

Suponga que A es una variable de tipo entera y que se desea asignarle un valor. Este valor será introducido por el usuario.

**scanf** (“%i%\*c”,  &var\_a);

El siguiente segmento de código declara una variable de tipo entera, despliega un mensaje en la pantalla de texto y lee el valor de una variable.

int A;   
**printf** (“Da un valor entero ->   ”);   
**scanf** (“%i%\*c”,  &var\_a);

**Nota:** La lectura de una variable siempre esta antecedida por el caracter &. Si este caracter se omite la lectura no tiene ningún efecto.

# Si deseas saber más puedes consultar:

(Joyanes Aguilar, L. y Zahonero Martínez, I. *Programación en C, C++, Java y UML*. México : McGraw Hill, 2010.) las siguientes secciones:

* Los elementos de un programa en C. (pp. 74)
* Tipos de datos en C. (pp. 76)
* El tipo de dato lógico. (pp. 80)
* Constantes. (pp. 81)
* Variables. (pp. 85)
* Duración de una variable. (pp. 87)
* Entradas y salidas. (pp. 89)
* Operadores y expresiones. (pp. 101)
* Operador de asignación. (pp. 101)
* Operadores aritméticos. (pp. 102)
* Operadores de incrementación y decrementación. (pp. 105)
* Conversión de tipos. (pp. 118)