LABORATORIO DE COMPUTACIÓN 1

Unidad 2 – Desarrollo y ejecución de un programa

VARIABLES Y CONSTANTES

Una variable representa un espacio de memoria para almacenar un valor de un determinado tipo.

<tipoVariable> <nombreVariable>;

```
int total;
int edad = 26;
```

Una constante también hace que el compilador reserve un espacio de memoria para almacenar un valor, pero en este caso ese dato es siempre el mismo y no se puede modificar durante la ejecución del programa.

```
const int IVA = 21;
```

REGLAS PARA EL USO DE VARIABLES

La línea

```
int total; /* sentencia de declaración */
```

es una **declaración** de la variable total, que indica al compilador que total es de tipo int (entero).

La línea

```
total = 160; /* sentencia de asignación */
```

es una **asignación**; sirve para dar (asignar) a la variable total el valor 160.

El valor que la variable almacena puede cambiar en la ejecución de un programa, pero el nombre sigue siendo el mismo.

```
total = 185;
```

REGLAS PARA EL USO DE VARIABLES

- Todas las variables que se utilicen en un programa escrito en C **deben ser declaradas previamente.**

- El valor de una variable que sólo ha sido declarada (sin asignarle un valor) **no esta definido**.

int contador;

NO puede suponerse que tenga el valor 0.

TIPOS DE DATOS

NUMÉRICOS ENTEROS

Nombre	Tamaño	Alcance
short	16 bits	desde −32.768 hasta 32.767
unsigned short	16 bits	desde 0 hasta 65.535
int	32 bits	desde −2.147.483.648 hasta 2.147.483.647
unsigned int	32 bits	desde 0 hasta 4.294.967.295
long	32 bits	desde −2.147.483.648 hasta 2.147.483.647
unsigned long	32 bits	desde 0 hasta 4.294.967.295
long long	64 bits	desde −9.223.372.036.854.775.808 hasta 9.223.372.036.854.775.807
unsigned long long	64 bits	desde 0 hasta 18.446.744.073.709.551.615

NUMÉRICOS REALES

Nombre	Tamaño	Alcance
float	32 bits	desde 1,17549e-38 hasta 3,40282e+38
double	64 bit	desde 2,22507e-308 hasta 1,79769e+308
long double	80 bits	desde 1,68105e-4932 hasta 1,18973e+4932

ALFANUMÉRICOS

El lenguaje C ofrece el tipo `char` para representar caracteres individuales (las letras mayúsculas y minúsculas, números del 0 al 9, signos de puntuación, etc.). Por ejemplo, la variable letra es de tipo `char`:

```
char letra;
letra = 'a';
```

SENTENCIAS Y BLOQUES

Las sentencias describen acciones algorítmicas que pueden ser ejecutadas.

Para C cada sentencia termina con un “;” (punto y coma)

Las sentencias se ponen unas debajo de otras, aunque sentencias cortas pueden colocarse en una misma línea.

EJEMPLOS

```
int nro = 15;
printf("Hola Mundo\n");
```

SENTENCIAS Y BLOQUES

Un bloque de código es un grupo de sentencias.

Un bloque de código está limitado por las llaves de apertura “{“y cierre “}”.

EJEMPLO

```
int main()
{
        linea1;
        linea2;
}
```

OPERADORES Y EXPRESIONES

Un operador es un símbolo formado por uno o más caracteres que permite realizar una determinada operación entre uno o más datos y produce un resultado.

Sirven para procesar variables y constantes.

Indica al compilador que debe efectuar una operación matemática o lógica.

Una expresión es un conjunto de datos unidos por operadores que tiene un único resultado

- Expresiones aritméticas
  - El resultado es un número
  - *a = ((2+6) / 8) * 3*
- Expresiones lógicas
  - El resultado es un valor verdadero o falso
  - *(a < 10) y (b > 50)*

TIPOS OPERADORES

OPERADORES ARITMÉTICOS

Suma (+); Resta (-); Producto (*); División (/) y Módulo (%)

```
int a = 20;
int b = 45;
int suma = a + b;
int resta = a - b;
int producto = a * b;
int division = a / b;
int modulo = b % a; //devuelve el resto de la división
                    //5 en este caso
```

OPERADORES RELACIONALES

Dan dos resultados posibles VERDADERO o FALSO

SIMBOLO	DESCRIPCION	EJEMPLO
<	menor que	(a < b)
>	mayor que	(a >b)
< =	menor o igual que	(a < = b)
>=	mayor o igual que	(a >>= b)
= =	igual que	(a = = b)
! =	distinto que	(a != b)

OPERADORES LÓGICOS

SIMBOLO	DESCRIPCION	EJEMPLO
&&	Y (AND)	(a>b) && (c < d)
||	O (OR)	(a>b) || (c < d)
!	NEGACION (NOT)	!(a>b)

OPERACIONES DE ASIGNACIÓN

Para realizar asignaciones se utiliza el operador =

EJEMPLO

La expresión y = x asigna a la variable ‘y’ el valor de la variable ‘x’

La expresión z = y = x asigna a las variables ‘z’ e ‘y’ el valor de ‘x’ (el operador = es asociativo por la derecha)

```
int x;
int y;
int z;

x = 65;
y = x; //asigna el valor de 'x' a la variable 'y'
z = y = x; //asigna el valor de 'x' a las
           //variables 'y' y 'z'
```

OPERADORES COMPUESTOS

+= y -=

EJEMPLO

Las dos líneas siguientes son equivalentes

```
temperatura = temperatura + 15;     // Sin usar asignación compuesta
temperatura += 15;     // Usando asignación compuesta
```

OPERADORES COMPUESTOS

Otros operadores de asignación son los de incremento(++) y decremento (--)

Estos operadores permiten, respectivamente, aumentar y disminuir **en una unidad** el valor de la variable sobre el que se aplican.

EJEMPLO Las tres líneas siguientes son equivalentes

```
temperatura = temperatura + 1; //Sin asignación compuesta ni incremento
temperatura += 1; //Usando asignación compuesta
temperatura++; //Usando incremento
```

OPERADORES COMPUESTOS

Si el operador ++ se coloca tras el nombre de la variable devuelve el valor de la variable antes de incrementarla, mientras que si se coloca antes, devuelve el valor de ésta tras incrementarla; lo mismo ocurre con el operador --.

EJEMPLO

```
c = b++; // Se asigna a c el valor de b y luego se incrementa b
c = ++b; // Se incrementa el valor de b y luego se asigna a c
```

TIPOS DE COMENTARIOS

```
//Esto es un comentario de una línea

/* Esto es un comentario de más de
una línea. */
```

IMPRESIÓN EN CONSOLA

```
printf("Hola Mundo");

printf("Hola mundo\n");
// \n imprime un salto de línea

printf("\ttab");
// \t deja espacio de tabulación antes de
//imprimir

printf("\n\t\tcombinado");
// combinando los modificadores.
```

LEER DATOS DE LA CONSOLA

```c
#include <stdio.h> //libreria que permite hacer uso de las funciones printf y scanf
#include <stdlib.h> //libreria que permite hacer uso de la función system

/* Ejemplo de escritura en consola y
    lectura de un valor ingresado por teclado */
int main()
{
    char inicialNombre;  // variable donde guardaremos la inicial ingresada
    int edad; // variable donde guardaremos la edad ingresada

    printf("Ingresa la inicial de tu nombre:\n");
    scanf("%c", &inicialNombre);

    printf("Ingresa tu edad:\n");
    scanf("%i", &edad);

    printf("Hola %c, tu edad es %i. \n",inicialNombre, edad);

    system("pause"); //necesario para que la consola no se cierre
}
```

LEER DATOS DE LA CONSOLA

El código %d, %c, etc. se usan dentro de las cadenas de control de las funciones printf y scanf, para indicar que el respectivo siguiente argumento ha de ser interpretado por el compilador como un entero, como un carácter, etc.

%i y %d - Para variables de tipo int.
%c - Para variables de tipo char.
%f - Para variables de tipo float.
%fl - Para variables de tipo double.