PROGRAMACIÓN I - UNIDAD 2: DATOS Y OPERACIONES

Autor: Ing.Florencia Marina Anderson

Contenido

DATOS 2
IDENTIFICADORES 2
NOMENCLATURAS DE NOMBRES DE IDENTIFICADORES 3
Buenas prácticas para la asignación de un identificador: 3
TIPOS DE DATOS 4
EXPRESIONES 5
OPERACIONES 7
Lectura 7
Escritura 8
Asignación 9
CONTADORES Y ACUMULADORES 10
BANDERA 11

Control de versiones:

Versión	Descripción cambios	Fecha
1.0	Inicial	
1.1	Se cambió camel case	2/5/2021

DATOS

Los datos son objetos que, al ser manipulados por diferentes sentencias o instrucciones de un programa, se convierten en información que ofrece dicho programa.

Un dato dentro de un programa se caracteriza por llevar asociado un identificador, un tipo y un valor.

- Identificador: Nombre para referenciar al dato dentro del programa
- Tipo: el tipo de un dato determina el rango de valores que puede tomar el dato y su ocupación en memoria durante la ejecución del programa
- Valor: Sera un elemento determinado dentro del rango de valores permitidos por el tipo de dato definido.

Algunos ejemplos de datos son: la edad, el saldo de una cuenta bancaria, el nombre de una persona, la letra del piso de una dirección, etc.

IDENTIFICADORES

Si se parte del hecho de que programar es solucionar problemas a través de variables, entonces un programa tiene algunas variables, pero también existen constantes.

Cada variable, constante e incluso algún proceso requieren un nombre que los identifique, y de ahí la palabra "identificadores", por lo que puede decirse que un identificador es el nombre de cualquier variable, constante, procedimiento o programa.

Variable: Una variable puede tomar distintos valores a lo largo de la ejecución de un programa. Por ejemplo, si tenemos una variable llamada `char_count` que registra la cantidad de caracteres que escribe un usuario, entonces el valor de ésta variable va a ir variando a medida que el usuario escribe o borra caracteres.

Constante: Una constante posee un solo valor fijo en toda la ejecución del programa. Supongamos que tenemos que desarrollar un método que recibe como entrada el radio de un círculo y debe calcular el área del mismo. Entonces, podríamos definir una constante `PI` con un valor fijo de 3,14 ya que como todos sabemos el valor de pi no varía. En cambio, vamos a definir una variable llamada `radio` y otra llamada `area` para el dato de entrada y el dato de salida que sabemos que van a variar cada vez que llamemos a ese método pasándole un radio distinto.

NOMENCLATURAS DE NOMBRES DE IDENTIFICADORES

La escritura de un identificador tiene reglas, las cuales deben aplicarse siempre en el momento de crear alguno de los elementos descritos.

"Un identificador debe iniciar con una letra, puede estar seguido de letras y/o números, no debe tener espacios en blanco, ni caracteres especiales, excepto el guion bajo (_) o el guion medio (-)"

Actualmente existen muchos convenios para asignar un nombre apropiado al identificador.

Buenas prácticas para la asignación de un identificador:

1. **El nombre debe ser representativo**

   Ejemplo:

   No es lo mismo

   ```
   x = 2 * y * z;
   ```

   Que:

   ```
   perimetro = 2 * PI * radio;
   ```

2. **No utilizar caracteres especiales (excepto "-" y "_")**

   Ejemplo:

   Usar el identificador `area` en vez del identificador `área` que lleva una tilde

3. **Empezar siempre con una letra**

   Ejemplo:

   Correcto: `total1`

   Incorrecto: `1total`

4. **Si el identificador consta de dos palabras, entonces utilizar mayúsculas para separar las palabras, o bien "_" o bien "–"**

   Ejemplos:

   TotalCost (convención PascalCase)

   totalCost (convención camelCase)

   total-cost (convención kebab-case)

   total_cost (convención snake_case)

TOTAL_COST (convención UNDER_SCORE)

Importante: el convenio que utilicemos, dependerá mucho del programador y también de la limitación del lenguaje, lo importante es siempre usar la misma convención para todo el código.

Sugerido para C: utilizar snake_case para las variables y UNDER_SCORE para las constantes

TIPOS DE DATOS

Estándares

- **Real (Float o Double en C):** Ejemplo, el saldo de una cuenta bancaria
- **Entero (Int en C):** Ejemplo, la edad de una persona
- **Caracter (Char en C):** Representa un único caracter. Ejemplo, podemos usarlo para identificar el estado civil de una persona, si el dato guarda solo los caracteres 'c','d','s','v'
- **Cadena de texto (Char[] en C):** Ejemplo el nombre de una persona
- **Lógico (Bool en C):** Los valores siempre indican o verdadero o falso. Ejemplo se puede usar para identificar si una persona es estudiante o no

Estructura definida por el programador

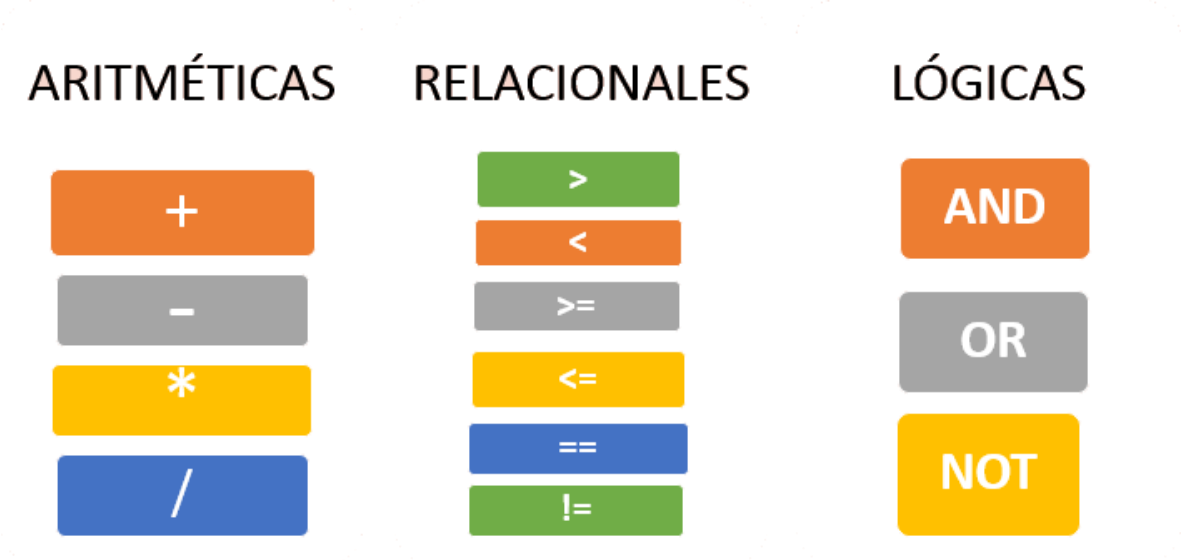
Una estructura puede ser definida por el progrmador y contiene varios datos. Vemos un ejemplo en C.

```
struct persona
{
    char[80] nombre;
    char[80] apellido;
    int edad;
};
```

EXPRESIONES

Un programa es un conjunto de instrucciones, y cada una de éstas, puede considerarse como una expresión, que no es más que la combinación de variables, constantes y operadores.

Dependiendo del tipo de operador, se clasifican en:



Ejemplo 1 de expresión aritmética: x=5+2

Ejemplo 2 de expresión aritmética: x = (-b) + (b^2 - 4*a*c)^(1/2)) / (2*a)

Ejemplo 1 de expresión relacional: 5<4 (el resultado de esto es FALSO)

Ejemplo 2 de expresión relacional: 5!=4 (el resultado de esto es VERDADERO)

Ejemplo 1 de expresión lógica: 4!=5 and 3!=2 (VERDADERO y VERDADERO → Resultado VERDADERO)

Ejemplo 2 de expresión lógica: 4!=5 and 3!=3 (VERDADERO y FALSO→ resultado FALSO)

Ejemplo 3 de expresión lógica: NOT 4!=5 (VERDADERO negado → resultado FALSO)

Ejemplo 4 de expresión lógica: NOT 4==5 (FALSO negado → resultado VERDADERO)

Ejemplo 5 de expresión lógica: 4==5 or 3==3 (FALSO o VERDADERO)→ resultado VERDADERO)

Nota:

- *AND: para que el resultado sea verdadero todos los elementos deben ser verdaderos, caso contrario (al menos uno falso), el resultado es falso.*
- *OR: para que el resultado sea verdadero al menos un elemento debe ser verdadero, caso contrario (todos falsos), el resultado es falso.*
- *NOT: invierte, si el elemento es verdadero el resultado es falso, y si el elemento es falso el resultado es verdadero.*

¿Qué sucede cuando tenemos una expresión del tipo?: ***A*** *OR* ***B*** *AND* ***C*** *OR* ***D***

En este caso primero se resuelven los AND y luego los OR.

Es decir: ***A*** *OR (****B*** *AND* ***C)*** *OR* ***D***

De todas formas, para evitar confusiones se prefiere el uso de paréntesis siempre en estos casos.

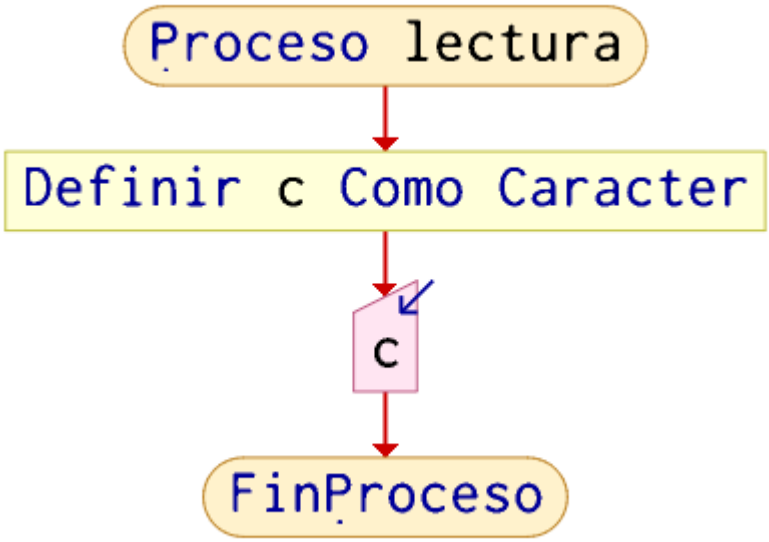
OPERACIONES

Una vez que se comprende el concepto de variable y/o constante, sus correspondientes tipos de datos y las expresiones que pueden escribirse al combinar operadores y operandos es posible realizar operaciones, es decir acciones que conduzcan a disponer de variables con valores que pueden obtenerse ya sea por parte de los usuarios de los programas o a su vez como resultado de la ejecución de una expresión.

Lectura

La operación de lectura, significa que una variable tomará un valor que será ingresado por el usuario a través de un dispositivo externo de entrada.

Ejemplo en diagrama de flujo:



Ejemplo en pseudocódigo:

```
Proceso lectura
        Definir fecha Como Caracter
        leer fecha
FinProceso
```

Ejemplo en C:

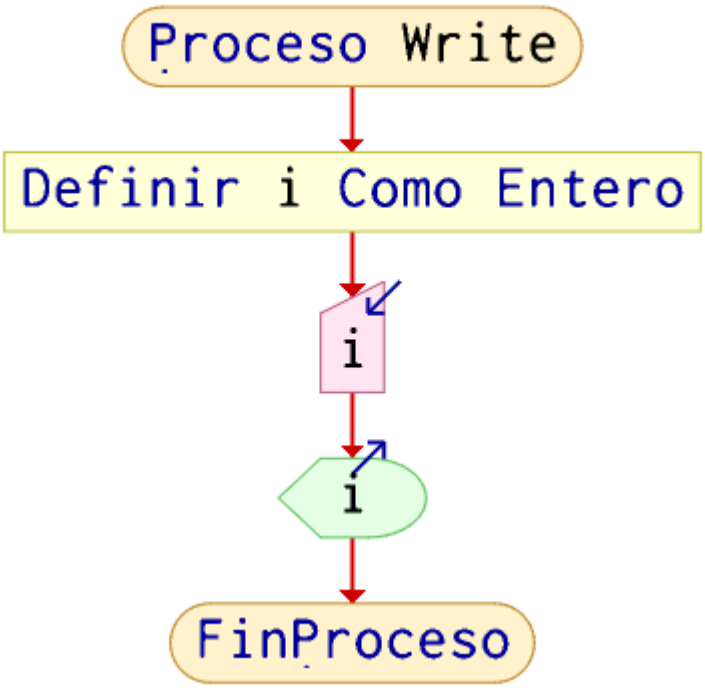
```
#include <stdio.h>

int main()
{
   char c; // definimos el tipo de variable
   c =getchar(); // le asignamos a la variable el valor que indique el usuario
}
```

Escritura

La operación de escritura expresa la necesidad de visualizar el valor que tiene una variable a través de un dispositivo de salida, ya sea una pantalla en la mayoría de casos, o una impresora.

Ejemplo en diagrama de flujo:



Ejemplo en pseudocódigo:

```
Proceso Write
        Definir i Como Entero
        Leer i
        Escribir i
FinProceso
```

Ejemplo en C:

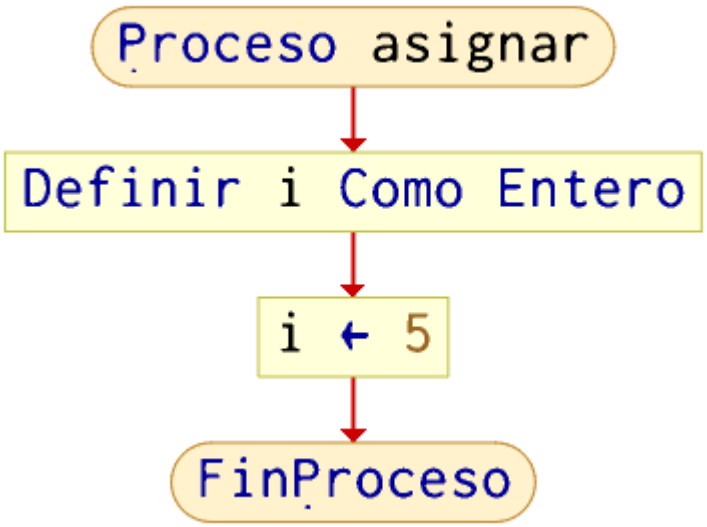
```
#include<stdio.h>

int main() {
        int i;
        scanf("%i",&i);
        printf("%i\n",i);
}
```

Asignación

Esta operación es una manera diferente que se utiliza para que una variable reciba un valor de forma directa sin intervención del usuario, o como resultado de la evaluación de una expresión. Para ésta operación se utiliza un operador que varía de acuerdo al lenguaje, pero que de forma básica puede ser una flecha con dirección hacia la izquierda.

Ejemplo de asignar el valor 5 a una variable entera en diagrama de flujo:



Ejemplo de asignar el valor 5 a una variable entera en pseudocódigo:

```
Proceso asignar
        Definir i Como Entera
        i ← 5
FinProceso
```

Ejemplo de asignar el valor 5 a una variable entera en C:

```
#include<stdio.h>
int main() {
        int i;
        i = 5;
}
```

CONTADORES Y ACUMULADORES

Contador: Variable que sirve para llevar registro de la cantidad de ocurrencias de un suceso (contador).

Ejemplo: Se tiene un conjunto de números y se necesita saber qué cantidad de éstos son números primos. Entonces, a medida que el sistema va analizando cada elemento del conjunto, cuando encuentra que existe un primo, incrementa en 1 el contador. Al final el contador tendrá como valor la cantidad de números primos de ese conjunto.

Tener en cuenta que para que ésto funcione antes de comenzar a iterar en cada elemento del conjunto, el valor del contador tiene que ser 0. A éste proceso de darle un valor inicial al contador se le llama **inicializar**

Acumulador: Variable que sirve para acumular valores.

Ejemplo: Se tiene un conjunto de notas de los alumnos de una carrera y se quiere sacar el promedio, entonces el sistema va recorriendo uno por uno cada elemento del conjunto y en cada iteración, un contador va registrando la cantidad de notas procesadas hasta el momento, y un acumulador va sumando la nueva nota a la suma que se fue haciendo de las anteriores. Cuando el proceso finalice, en el contador tendremos la cantidad de notas y en el acumulador la suma de todas y con éstos datos podremos calcular el promedio.

Entrada: Conjunto de notas N= {6, 8, 7} valor inicial del contador=0 valor inicial del acumulador=0

Paso	Elemento analizado	Valor del elemento	Contador= Contador +1	Acumulador=Acumulador + nota
0	ninguno		0	0
1	Primer elemento	6	1	6

2	Segundo elemento	8	2	14
3	Tercer elemento	7	3	21

Al finalizar la iteración el contador=3 y el acumulador=21 por lo que concluimos que el promedio es 21/3=7

BANDERA

En programación, la bandera o flag es una variable lógica (booleana) que nos indica si ha ocurrido un suceso.

Ésta variable cambia entre dos posibles valores (Verdadero/Falso, Si/No, 1/0), ya sea para alternar la ejecución de dos bloques de código o para producir la salida de un ciclo por la acción de que la variable tenga o no uno de sus valores.

Ejemplo:

Se necesita analizar si un conjunto de palabras ingresadas por el usuario contiene al menos un palíndromo (palabra que se lee igual de izquierda a derecha o de derecha a izquierda).

Entonces, como entrada tenemos un conjunto de n palabras.

Creamos una bandera llamada **existe_palindromo** que en un principio va a tener el valor falso (inicialización).

El sistema comienza a analizar una a una cada palabra del conjunto, cuando encuentra un palíndromo, entonces a la bandera **existe_palindromo** le da un valor verdadero. Y el sistema ya definió que cuando ésta bandera tenga valor verdadero ya no analizará el resto de las palabras y dejará de iterar para cada elemento del conjunto (ya que con que haya al menos un palíndromo es suficiente para la solución que se pretende dar).

Si el sistema analizó todas las palabras y no encontró ningún palíndromo entonces la bandera quedará con el valor inicial que se le dio **existe_palindromo=Falso**