Presentación ***resumen*** del libro:

"EMPEZAR DE CERO A PROGRAMAR EN lenguaje C"

Autor: Carlos Javier Pes Rivas (correo@carlospes.com)

Capítulo 5

IDENTIFICADORES, VARIABLES Y CONSTANTES

OBJETIVOS

- Saber qué es y para qué sirve un **identificador**.
- Saber qué son y para qué sirven las **variables** y las **constantes**.
  - En el capítulo anterior se han estudiado algunos de los tipos de datos que pueden utilizar los programas, pero, ¿dónde están los datos que utiliza un programa?

CONTENIDO

5.1 INTRODUCCIÓN

- Para **diseñar algoritmos** en pseudocódigo, se pueden utilizar los siguientes elementos:

  - **Tipos de datos**
  - **Variables**
  - **Constantes**
  - **Operadores**
  - **Expresiones**
  - **Instrucciones**

5.2 IDENTIFICADORES (1/4)

- **Un identificador:** es el ***nombre*** que se le da a un elemento de un algoritmo o (programa).

- **Identificadores predefinidos:**
  - **EJEMPLO:**
    - **`entero`**, **`real`**, **`logico`**, **`caracter`**, **`cadena`**, **`verdadero`** y **`falso`** forman parte del lenguaje algorítmico (son **palabras reservadas**).

- **Identificadores definidos por el programador:**
  - **EJEMPLOS:**
    - Variables, constantes,...
    - Nombre de algoritmos.

5.2 IDENTIFICADORES (2/4)

- **Reglas de sintaxis de un identificador:**
  - En nuestro **pseudocódigo CEE** (***C En Español***) vamos a usar las mismas reglas de sintaxis que existen en C.

  1. Consta de uno o más caracteres.
  2. El primer carácter debe ser una letra o el carácter *subrayado* (_), mientras que, todos los demás pueden ser letras, dígitos o el carácter *subrayado* (_). Las *letras* pueden ser minúsculas o mayúsculas del alfabeto inglés. Así pues, no está permitido el uso de las letras ‘ñ’ y ‘Ñ’.
  3. No pueden existir dos identificadores iguales, es decir, dos elementos de un algoritmo no pueden nombrarse de la misma forma. Lo cual no quiere decir que un identificador no pueda aparecer más de una vez en un algoritmo.

  - **Las vocales no pueden llevar tilde ni diéresis.**

5.2 IDENTIFICADORES (3/4)

- **EJEMPLOS:** Identificadores **válidos** definidos por el programador son:

```
numero
dia_del_mes
PINGUINO1
_ciudad
Z
```

- **EJEMPLOS:** Identificadores **no válidos** son:

```
123
_DÍA
numero*
lugar de nacimiento
año
```

5.2 IDENTIFICADORES (4/4)

- **EJEMPLOS:** **No pueden** ser definidos por el programador:

  ```
  entero
  logico
  ```

  – Son identificadores predefinidos (ya existen).

- **EJEMPLOS:** Identificadores **válidos** y **distintos** son:

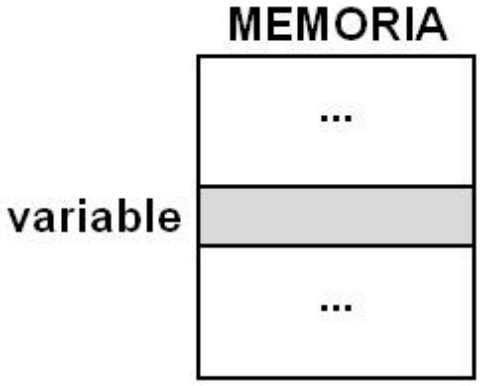
  ```
  Mes
  mes
  ```

  – Los identificadores son sensibles a minúsculas y mayúsculas.

- Es aconsejable que los identificadores tengan un significado afín a lo que representan.

5.3 VARIABLES (1/4)

- **Variable:**
  - Representa a un **espacio de memoria** en el cual se puede almacenar un dato.
  - El programador, cuando desarrolla un programa (o diseña un algoritmo), debe decidir:
    - Cuantas son las variables que el programa necesita para realizar las tareas que se le han encomendado.
    - El tipo de dato que puede tomar cada una de ellas.
  - Durante la ejecución de un programa, **el valor** que tome el dato almacenado en una variable **puede cambiar** tantas veces como sea necesario.
  - El **tipo de dato** de una variable **no puede ser cambiado** durante la ejecución de un programa.
  - Gráficamente, se puede representar como:



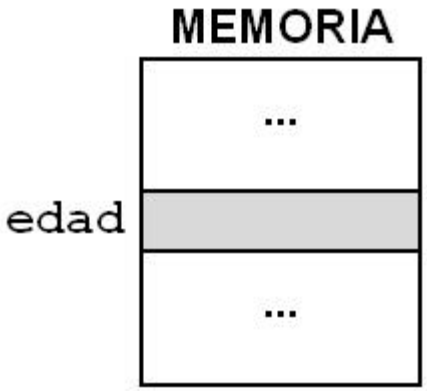
5.3 VARIABLES (2/4)

- **Declaración de variables:** Para que un programa pueda hacer uso de una o más variables, éstas deben ser declaradas previamente, indicando de cada una de ellas:

  1. El **tipo de dato** que puede almacenar (mediante un **identificador**).
  2. Su **nombre** (mediante otro **identificador**).

- **EJEMPLO:** variable para almacenar la edad de una persona:

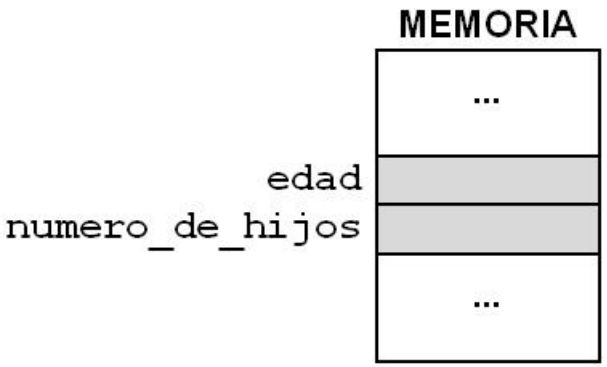
```
entero edad
```



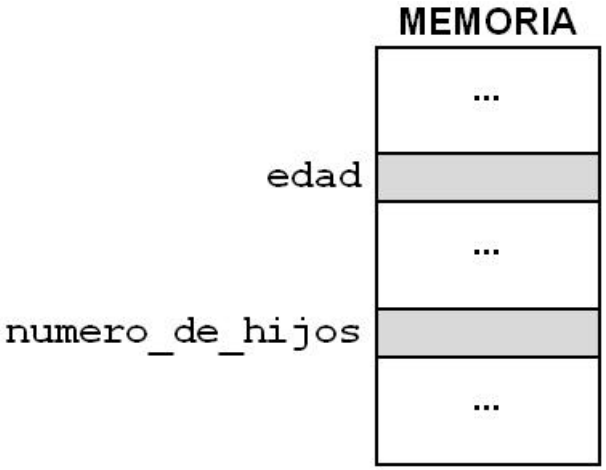
5.3 VARIABLES (3/4)

- **EJEMPLO:** variables para almacenar la edad de una persona y su número de hijos:

```
entero edad
entero numero_de_hijos
```



  - Las variables no tienen por qué estar contiguas en la memoria.



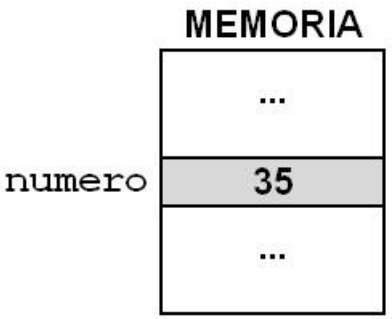
  - Puesto que las dos variables son del **mismo tipo de dato**, se pueden declarar en la misma línea separándolas por medio de una coma (,).

```
entero edad, numero_de_hijos
```

5.3 VARIABLES (4/4)

- **EJEMPLO:** Opcionalmente, cuando se declara una variable, a ésta se le puede asignar un **valor inicial**.

```
entero numero = 35
```



- **Sintaxis para declarar una variable:**

```
<nombre_del_tipo_de_dato> <nombre_de_la_variable> [ = <expresión> ]
```

- **Sintaxis para declarar más de una variable:**

```
<nombre_del_tipo_de_dato> <nombre_de_la_variable_1> [ = <expresión_1> ],
                          <nombre_de_la_variable_2> [ = <expresión_2> ],
                           ...,
                          <nombre_de_la_variable_n> [ = <expresión_n> ]
```

5.4 CONSTANTES (1/8)

- **Constante:**
  - Representa a un **valor** (dato almacenado en memoria) que **no puede cambiar** durante la ejecución de un programa.
  - En C, una constante puede ser de tipo entero, real, carácter, cadena o enumerado.
  - Las constantes de tipo enumerado se van a estudiar en el capítulo 6. En cuanto a las demás, se pueden expresar de dos formas diferentes:
    1. Por su **valor**.
    2. Con un **nombre** (**identificador**).

- **EJEMPLOS:** constantes expresadas por su valor:

```
-5
10
```

5.4 CONSTANTES (2/8)

- Para expresar una constante con su nombre, la constante debe ser declarada previamente, indicando:
  1. Su **nombre** (mediante un **identificador**)
  2. El **valor** que simboliza (mediante una **expresión**)
- En pseudocódigo, para declarar una constante, vamos a utilizar la sintaxis:

```
<nombre_de_la_constante> = <expresión>
```

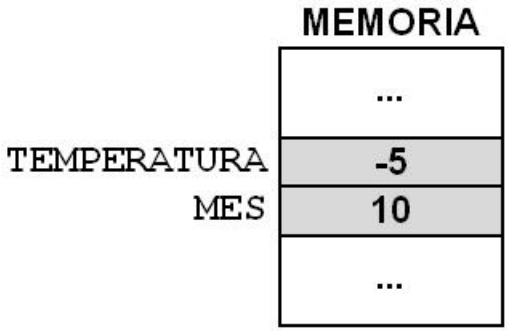
- **EJEMPLOS:** constantes expresadas por su valor:

```
TEMPERATURA = -5

MES = 10
```

- Para declarar más de una constante en una misma línea, las separaremos por medio de comas (,).

```
TEMPERATURA = -5, MES = 10
```



5.4 CONSTANTES (3/8)

- **Constante de tipo entero:** Representa a un **valor** (dato) perteneciente al subconjunto de *Z* representable por el ordenador.
- **EJEMPLO:** Suponiendo que con 16 bits, en Complemento a 2, el ordenador puede representar { -32768,..., -1, 0, 1,... ,32767 }
  - **Correctos: dígitos del (0) al (9) y los caracteres (+) y (-)**
    `-32000`
    `0`
    `000077` (Los ceros a la izquierda no son significativos)
    `+1111`
  - **Incorrectos:**
    `-32.000` (No se puede usar el carácter punto (`.`))
    `0,0` (No se puede usar el carácter coma (`,`))
    `++111` (No se puede duplicar el signo)
    `38000` (No pertenece al subconjunto de *Z* representable)

5.4 CONSTANTES (4/8)

- **Constante de tipo real:** Representa a un **valor** (dato) perteneciente al subconjunto de *R* representable por el ordenador.
- **EJEMPLOS:**
  - **Correctos: dígitos del (0) al (9) y los caracteres (+), (-), (.), (e) y (E)**

    `8.2`

    `000.333` (Los ceros a la izquierda no son significativos)

    `+1111.809`

    `-3200.` (También se puede escribir `-3200.0`)

    `.56` (También se puede escribir `0.56`)

    `-77e-3`

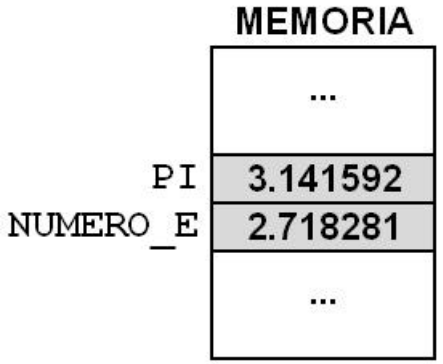
    `2000E+2`

    `3040e2`

    `-50.50e-4`

    `400.e-3`

    `.7e3`



5.4 CONSTANTES (5/8)

- **EJEMPLOS:**
  - **Incorrectos:**
    - `-200` (No aparece el punto ni el exponente)
    - `-20,0` (No puede aparecer la coma)
    - `--111.` (No se puede duplicar el signo)
    - `-111..` (No se puede duplicar el punto)
    - `-111.11.` (No puede aparecer más de un punto)
    - `+22e` (Después del carácter (**e**) o (**E**) se debe escribir el exponente)
    - `+22ee6` (No se puede duplicar el carácter (**e**) o (**E**))
    - `+22e 6` (No se puede escribir el carácter *espacio en blanco*)
    - `380E-2.2` (El exponente debe ser una cantidad entera)

5.4 CONSTANTES (6/8)

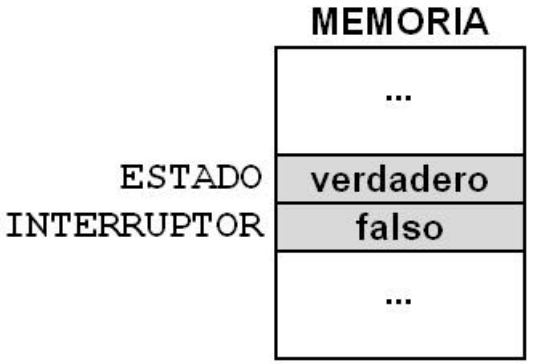
- **Constante de tipo lógico:** Representa a un **valor** (dato) perteneciente al conjunto:

  { **`verdadero`**, **`falso`** }

  – No suele ser habitual declarar constantes de este tipo de dato.
  – En C no existe el tipo de dato lógico.

- **EJEMPLOS:**

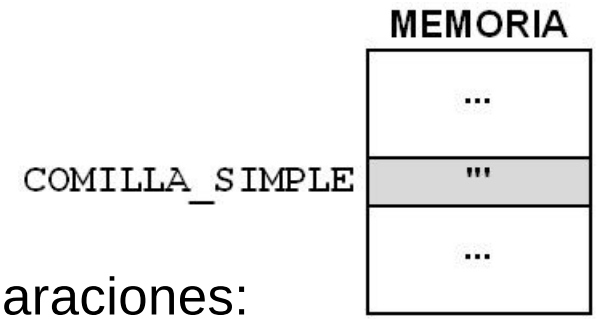
```
ESTADO = verdadero
INTERRUPTOR = falso
```

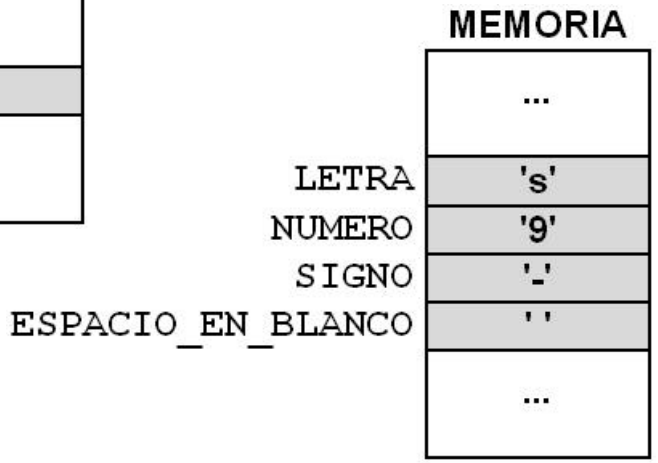


5.4 CONSTANTES (7/8)

- **Constante de tipo carácter:** Representa a un **valor** (dato) perteneciente al conjunto de caracteres que puede representar el ordenador.
- **EJEMPLOS:** constantes de tipo carácter expresadas por su valor:

```
'a'
'T'
'5'
'+'
'.'
```





- **EJEMPLOS:** declaraciones:

```
LETRA = 's'
NUMERO = '9'
SIGNO = '-'
ESPACIO_EN_BLANCO = ' '
COMILLA_SIMPLE = '\''
```

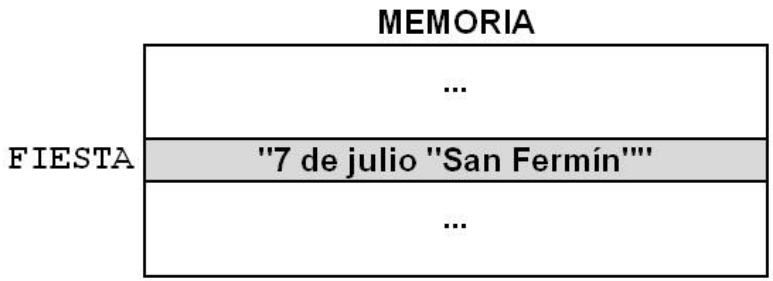
5.4 CONSTANTES (8/8)

- **Constante de tipo cadena:** Representa a un **valor** (dato) de tipo cadena, es decir, representa a una secuencia de caracteres.
- **EJEMPLOS:** constantes de tipo cadena expresadas por su valor:

```
"Alejandro"
"Lucerna"
"Barcelona 2000"
```

- **EJEMPLOS:** declaraciones:

```
NOMBRE = "Alejandro"
CIUDAD = "Lucerna"
OLIMPIADAS = "Barcelona 2000"
FIESTA = "7 de julio \"San Fermín\""
```



EJERCICIOS RECOMENDADOS

- **Resueltos:** 1, 2 y 3.
- **Propuestos:** 1 y 2.

GRACIAS POR SU ATENCIÓN

Para más información, puede visitar la web del autor:

http://www.carlospes.com