

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Estructura de Datos y Algoritmos I

Actividad 2: Acordeón de Lenguaje C y Lenguaje TypeScript.

Torres Oropeza Diego Alberto

26/02/2021



# Lenguaje C

## Cosas básicas:

- “;” al terminar una línea.
- “//” para comentar una línea.
- “/\*, \*/” para comentar varias líneas.

## Librerías (algunas):

Al llamar alguna librería al código, siempre se pone un “#include” antes de la librería.

- <stdio.h>: Núcleo de todo código, funciones de entrada y salida de datos.
- <stdlib.h>: Funciones para acciones como controlar procesos, ordenar datos, buscar datos, etc.
- <conio.h>: Funciones para mejor la entrada y salida de datos en consola.
- <math.h>: Funciones matemáticas.
- <string.h>: Funciones para manipulación de cadenas de caracteres.
- <time.h>: Funciones para tratamiento y conversión de formatos de fecha y hora.

## Tipos de datos:

- Int: Entero, %i, %d.
- Float: Decimales, %f.
- Double: Decimales con mayor rango que float, %f.
- Char: Carácter, %c.

## Declarar datos:

- **Int** numero;
- **Float** decimal;
- **Char** letra;
- **Char** dia[10];

## Forma base de un código:

```
#include <stdio.h> //librería de cajón
```

```
Int main() //código principal
```

```
{
```

```
// Aquí va todo el código
```

```
}
```

## Operadores aritméticos:

- "+": Suma.
- "-": Resta.
- "\*": Multiplicación.
- "/": División.
- "++": Incremento.
- "--": Decremento.
- "%": Módulo.
- "+=": Suma una variable más un número u otra variable -> A+=C -> A=A+C
- "-=": Resta una variable o un número de otra -> A-=C -> A=A-C
- "\*=": Multiplica una variable por otra o por un número -> A\*=C -> A=A\*C
- "/=": Divide una variable entre otra o entre un número -> A/=C -> A=A/C

## Operadores Racionales:

- ">": Mayor que.
- "<": Menor que.
- ">=": Mayor o igual que.
- "<=": Menor o igual que.
- "==": Exactamente igual.
- "!=": Diferente.

## Operadores lógicos:

- "&&": and.
- "||": or.
- "!": not.

## Constantes de caracteres de barra invertida:

Se ponen al final de un dato de salida, dentro de las comillas, para hacer una acción que no se puede hacer con el teclado.

- \b: Retroceso.
- \n: Salto de línea.
- \t: Tabulador horizontal.
- \v: Tabulador vertical.
- \": Doble comilla.
- \': Comilla simple.

## Funciones útiles:

- `Printf()`: Salida de datos.
- `Scanf()`: Entrada de datos.
- `Gets()`: Entrada de datos tipo cadena.
- `Puts()`: Salida de datos tipo cadena.
- `Fclose()`: Cierra el archivo asociado a "f".
- `Fopen()`: Abre el archivo asociado a "f".
- `Error()`: Comprueba si existen errores en alguna operación realizada al archivo.
- `Feof()`: Comprueba el indicador de posición del archivo.
- `Fflush()`: Vacía el buffer asociado a "f".
- `Fgetc()`: Devuelve el carácter del archivo de entrada asociado a "f".
- `Fgets()`: Lee caracteres del archivo asociado a "f".
- `Fprintf()`: Escribe en el archivo asociado a "f" los valores de los argumentos dados.
- `Strcmp()`: Compara dos cadenas.
- `Strcpy()`: Copia el contenido de una cadena a otra.
- `Strlen()`: Devuelve el número de caracteres de la cadena.
- `Acosh()`: Devuelve el arcocoseno del argumento.
- `Asinh()`: Devuelve el arcoseno del argumento.
- `Atanh()`: Devuelve el arcotangente del argumento.
- `Cosh()`: Devuelve el coseno del argumento.
- `Cosh()`: Devuelve el coseno hiperbólico del argumento.
- `Exp()`: Número "e" elevado a la potencia del argumento.
- `Fabs()`: Devuelve el valor absoluto de un número.
- `Fmod()`: Devuelve la división entera de x/y.
- `Log()`: Devuelve el logaritmo neperiano de un número.
- `Log10()`: Devuelve el logaritmo base 10 de un número.
- `Pow()`: Devuelve la base elevada al exponente, es una potencia.
- `Sin()`: Devuelve el seno del argumento.
- `Sinh()`: Devuelve el seno hiperbólico del argumento.
- `Sqrt()`: Devuelve la raíz cuadrada del número.
- `Tan()`: Devuelve la tangente del argumento.
- `Tanh()`: Devuelve la tangente hiperbólica del argumento.
- `Abs()`: Devuelve el valor absoluto del entero dado por el número.
- `Exit()`: Terminación inmediata del programa.
- `Labs()`: Devuelve el valor absoluto del número.

## Estructura condicional “if”:

El “if” nos permite realizar una serie de acciones siempre y cuando se cumpla una condición, “else” se usa para cuando esa condición no se cumple, dar otra salida:

```
If(num==1)
{
Printf(“Hola!!”);
}else{
Printf(“Adios!!”);}
```

## Estructura de selección “switch-case”:

Esta estructura nos sirve para cuando, de acuerdo a qué datos de entrada el usuario le da a una variable, con esa variable se hagan ciertas acciones, al finalizar cada acción o serie de acciones que se deben cumplir si se elige esa opción, se coloca un “break” para indicar que ahí finaliza lo que cierta opción hace. Si no se cumple con ninguna de las opciones, “default” nos da una salida:

```
Switch(opcion)
Case 1:
    //serie de acciones
Break;
Case 2:
    //serie de acciones
Break;
Default:
    //acción para cuando no se elija alguna opción
```

## Estructura cíclica “for”

Para usar cada estructura cíclica, no sólo “for”, si no “while” y “do-while” también, se necesita un contador que nos permite llevar un conteo de los ciclos para así no tener ciclos específicos, una variable o un número son lo necesario para ponerle un tope al contador. Dentro del ciclo se debe considerar la serie de acciones a hacer y, muy importante, aumentar el valor del contador con cada vuelta:

```
For(i=0, i<=10,i++)
```

```
{
```

```
//serie de acciones
```

```
}
```

“i” es el contador, inicia con un valor de 0 y se hacen la serie de acciones del ciclo hasta que “i” sea igual o menor a 10. Como ya vimos antes, “++” incrementa el valor del contador en 1 unidad cada vez que se completa el ciclo para así no tener un ciclo infinito.

## Estructura cíclica “while”

Como en “for”, se necesita de un contador para no tener un ciclo infinito:

```
Int i=0;
```

```
While(i<=10)
```

```
{
```

```
//serie de acciones
```

```
i++;
```

```
}
```

Aquí se debe considerar que el incremento va dentro de las acciones del ciclo, para no olvidar ponerlo. También puede usarse el decremento y poner “while(i>=10)”, dándole un valor inicial a “i” mayor a 10.

## Estructura cíclica “do-while”

Como en “while”, el incremento al contador va dentro de la serie de acciones, pero ahora al inicio del ciclo y no al final por la forma de la estructura:

```
Do
{
i++;
//serie de acciones
}
While(i<=10);
```

## Arreglos

Los arreglos son como vectores que nos permiten almacenar diferentes datos en una sola variable, gracias a que esta variable tiene como que espacios para cada dato. Se pueden declarar arreglos de diferentes tipos de datos:

```
Int arreglo[4];
Char arreglo1[5];
Float arreglo2[6];
```

Los números dentro de los corchetes indican las casillas que tendrá el arreglo, en estos casos, cada arreglo podrá guardar como máximo 4,5 y 6 datos de su tipo, respectivamente.

## Matrices

Las matrices, como en si vio en Álgebra, son tablas de datos, con filas y columnas. En programación esas filas y columnas se representan con los corchetes, mientras que en los arreglos poníamos sólo 1 par de corchetes, para las matrices son 2:

```
Int matriz[2][2];
Char matriz1[3][3];
Float matriz2[4][4];
```

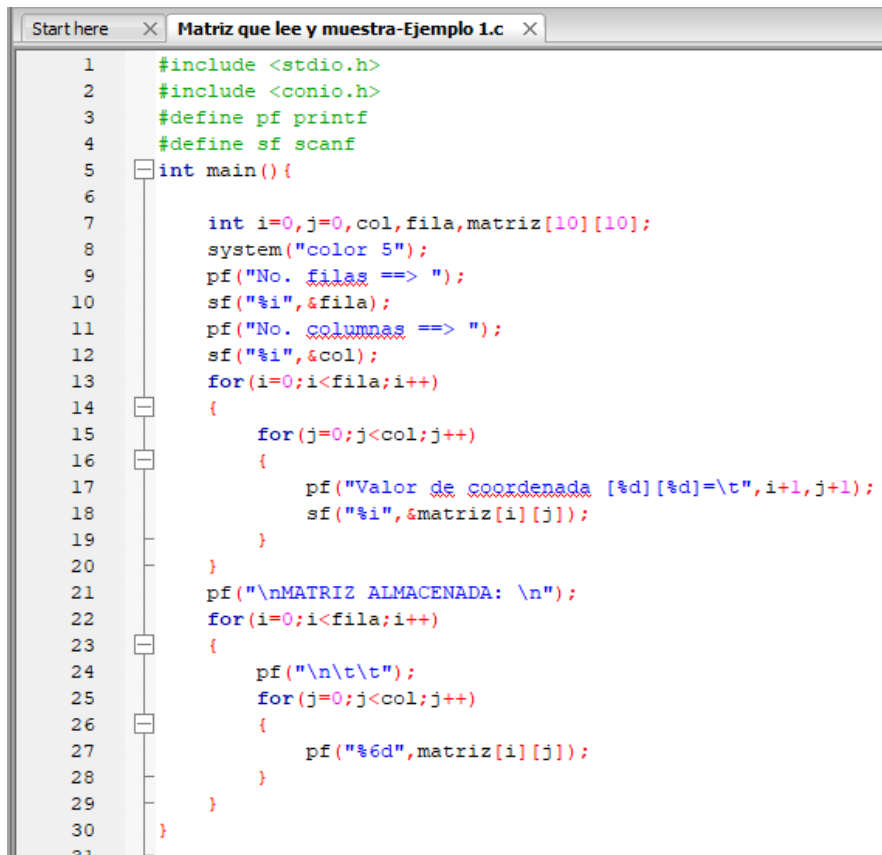
En este caso, cada matriz es una matriz cuadrada, cada una almacena en forma de tabla 4,9 y 16 datos, respectivamente.

## Método burbuja

El método burbuja es el que nos ayuda a llenar las matrices con los datos que se quieran introducir.

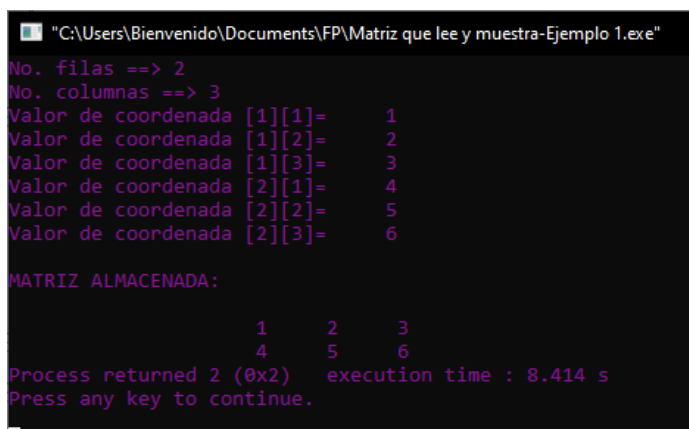
Consiste en meter un ciclo “for” dentro de otro ciclo “for” para poder llenar primero las columnas y luego las filas de la matriz o viceversa, como mejor se le acomode al programador o como se le pida. Para poder realizar el llenado de filas y columnas se usan 2 contadores “i” y “j” para no crear bucles infinitos.

A continuación presento un código aterrizando los conceptos de ciclos “for”, matrices, funciones para entrada y salida de datos y constantes de caracteres de barra invertida:



```
1  #include <stdio.h>
2  #include <conio.h>
3  #define pf printf
4  #define sf scanf
5  int main() {
6
7      int i=0,j=0,col,fila,matrix[10][10];
8      system("color 5");
9      pf("No. filas ==> ");
10     sf("%i",&fila);
11     pf("No. columnas ==> ");
12     sf("%i",&col);
13     for(i=0;i<fila;i++)
14     {
15         for(j=0;j<col;j++)
16         {
17             pf("Valor de coordenada [%d][%d]=\t",i+1,j+1);
18             sf("%i",&matrix[i][j]);
19         }
20     }
21     pf("\nMATRIZ ALMACENADA: \n");
22     for(i=0;i<fila;i++)
23     {
24         pf("\n\t\t");
25         for(j=0;j<col;j++)
26         {
27             pf("%6d",matrix[i][j]);
28         }
29     }
30 }
31
```

El ejecutable de este código se ve así:



```
"C:\Users\Bienvenido\Documents\FP\Matriz que lee y muestra-Ejemplo 1.exe"
No. filas ==> 2
No. columnas ==> 3
Valor de coordenada [1][1]= 1
Valor de coordenada [1][2]= 2
Valor de coordenada [1][3]= 3
Valor de coordenada [2][1]= 4
Valor de coordenada [2][2]= 5
Valor de coordenada [2][3]= 6

MATRIZ ALMACENADA:

    1    2    3
    4    5    6
Process returned 2 (0x2)   execution time : 8.414 s
Press any key to continue.
```



## Funciones

Las funciones nos permiten meter dentro de ellas una serie de acciones que se cumplen con la entrada de x dato y pueden o no devolver otro dato. Sirven principalmente para optimizar el código, ya que antes usábamos puras llaves ({} ) para separar secciones de código, pero con las funciones podemos meter todas esas secciones dentro de una función y en el código principal únicamente llamar a la función. Las funciones siempre van antes del programa principal, pero si quieres tener el principal de primeras, puedes declarar después de la llamada a librerías las funciones, poner el código "main" después eso y finalmente todas las funciones a usar previamente declaradas. A continuación presento un ejemplo realizado en la práctica de este tema, ya que explicar esto escribiendo está medio complicado:

```
Práctica12_Ejercicio4_MatrizTranspuesta.c - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Start here x Matriz que lee y muestra-Ejemplo 1.c x Práctica12_Ejercicio4_MatrizTranspuesta.c x

1  #include <stdio.h>
2  #include <conio.h>
3  #define pf printf
4  #define sf scanf
5  #define max 10
6
7  void fAcomodaDatos(int matriz[][10],int f, int c)///Traspuesta
8  {
9      int i,j;
10     for(i=0;i<c;i++)
11     {
12         pf("\n\t\t");
13         for(j=0;j<f;j++)
14         {
15             pf("%6d",matriz[j][i]);
16         }
17     }
18 }
19
20 void fImprimeDatos(int matriz[][10],int f, int c)///Imprime matriz original
21 {
22     int i,j;
23
24     for(i=0;i<f;i++)
25     {
26         pf("\n\t\t");
27         for(j=0;j<c;j++)
28         {
29             pf("%6d",matriz[i][j]);
30         }
31     }
32 }
33
34 void fGuardaDatos(int matriz[max][max], int f, int c)
35 {
36     int i,j;
37
38     for(i=0;i<f;i++)///Leer coords. de matriz
39     {
40         for(j=0;j<c;j++)
41         {
42             pf("Valor de coordenada [%d][%d]=\t",i+1,j+1);
43             sf("%i",&matriz[i][j]);
44         }
45     }///FIN LEER DATOS
46 }
```

```

47     int fLeeDatos()
48     {
49         int nl;
50         sf("%i",&nl);
51         return nl;
52     }
53
54     int main()
55     {
56         int col, filas;
57         system("color 1");
58         pf("No. FILAS-->");
59         filas=fLeeDatos();
60         pf("No. COLUMNAS-->");
61         col=fLeeDatos();
62         int matriz[filas][col];
63         printf("MATRIZ\n");
64         fGuardaDatos(matriz,filas,col);
65         pf("\nMatriz almacenada:\n");
66         fImprimeDatos(matriz,filas,col);
67         pf("\n");
68         pf("Matriz transpuesta:\n");
69         fAcomodaDatos(matriz,filas,col);
70         pf("\n");
71     }
72

```

Como podemos observar, primero van todas las funciones y finalmente el código “main”. En el código “main” se puede ver cómo en las funciones, dentro de los paréntesis, hay variables, esto indica que esas variables son las que necesita la función para trabajar. El ejecutable de este código se ve así:

```

C:\Users\Bienvenido\Documents\FP\PrBctica12_Ejercicio4_MatrizTranspuesta.exe
No. FILAS-->3
No. COLUMNAS-->3
MATRIZ
Valor de coordenada [1][1]=      1
Valor de coordenada [1][2]=      2
Valor de coordenada [1][3]=      3
Valor de coordenada [2][1]=      4
Valor de coordenada [2][2]=      5
Valor de coordenada [2][3]=      6
Valor de coordenada [3][1]=      7
Valor de coordenada [3][2]=      8
Valor de coordenada [3][3]=      9

Matriz almacenada:

      1      2      3
      4      5      6
      7      8      9

Matriz transpuesta:

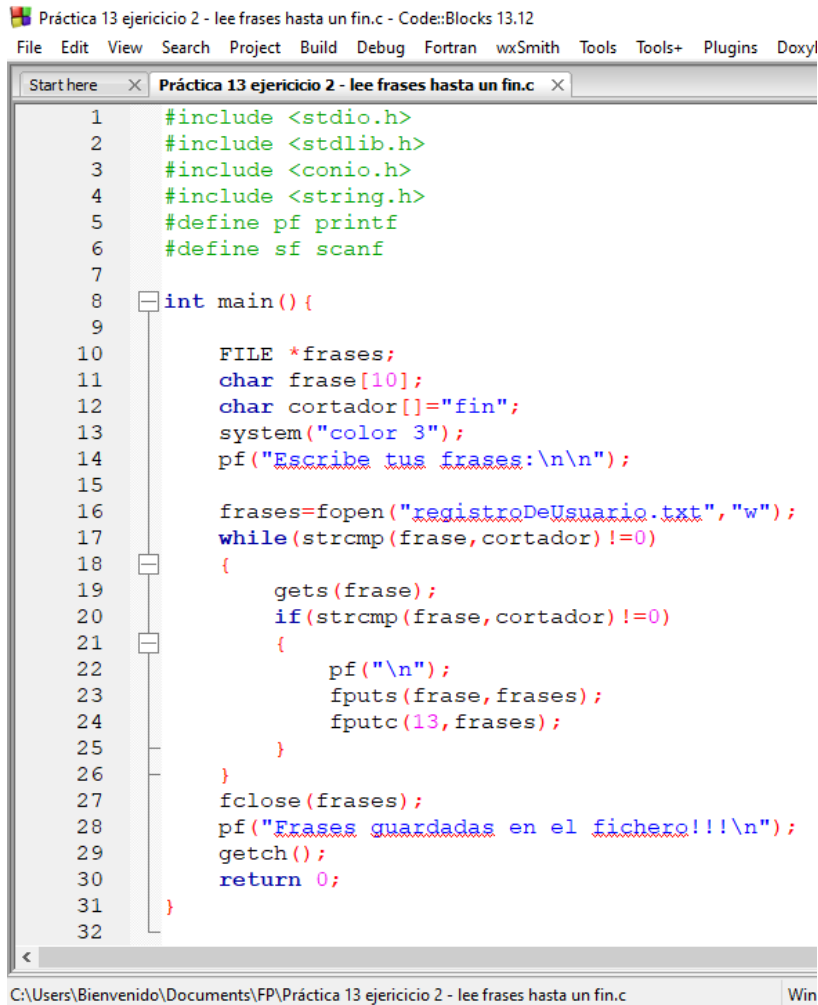
      1      4      7
      2      5      8
      3      6      9

Process returned 10 (0xA)   execution time : 7.400 s
Press any key to continue.

```

## Manejo de archivos

El manejo de archivos nos permite pasar información de un archivo al código en el ejecutable, o del ejecutable del código a un archivo. Las funciones que vimos antes como `fopen`, `fclose`, `fputs`, `fputc`, etc., son las que nos ayudan en la realización de estos códigos, a continuación, presento un código realizado en esta práctica y posteriormente lo explicaré:



The screenshot shows a code editor window titled "Práctica 13 ejercicio 2 - lee frases hasta un fin.c". The code is as follows:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4  #include <string.h>
5  #define pf printf
6  #define sf scanf
7
8  int main() {
9
10     FILE *frases;
11     char frase[10];
12     char cortador[]="fin";
13     system("color 3");
14     pf("Escribe tus frases:\n\n");
15
16     frases=fopen("registroDeUsuario.txt", "w");
17     while(strcmp(frase, cortador) != 0)
18     {
19         gets(frase);
20         if(strcmp(frase, cortador) != 0)
21         {
22             pf("\n");
23             fputs(frase, frases);
24             fputc(13, frases);
25         }
26     }
27     fclose(frases);
28     pf("Frases guardadas en el fichero!!!\n");
29     getch();
30     return 0;
31 }
32
```

The status bar at the bottom shows the file path: "C:\Users\Bienvenido\Documents\FP\Práctica 13 ejercicio 2 - lee frases hasta un fin.c" and the operating system: "Win".

Este código lo que hace es que, en el ejecutable, el usuario puede ir metiendo palabras al azar, de menos de 10 caracteres, y cuando escriba "fin", el ejecutable terminará, todas las frases introducidas, menos el "fin", se guardarán en un fichero de nombre "registroDeUsuario.txt".

También podemos ver en la línea 17 el uso de "strcmp" para comparar que la frase tecleada por el usuario no sea igual a "fin".

A continuación, veremos el ejecutable y si en realidad se pasan los datos a un archivo:

```
"C:\Users\Bienvenido\Documents\FP\PrBctica 13 ejercicio 2 - lee frases hasta un fin.exe"
Escribe tus frases:
casa
cobija
anaquel
musica
conejo
musgo
luna
cielo
alien
mesa
metal
tarea
Pedro
fin
Frases guardadas en el fichero!!!
```

```
registroDeUsuario: Bloc de notas
Archivo Edición Formato Ver Ayuda
|
casa
cobija
anaquel
musica
conejo
musgo
luna
cielo
alien
mesa
metal
tarea
Pedro
```

Hasta este tema llegaron las prácticas del curso de Fundamentos en Programación, espero haya sido de ayuda este pequeño repaso :D.

# Lenguaje TypeScript (tomado por mi apellido: Torres)

Es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Se usa para desarrollar aplicaciones JavaScript.

TypeScript amplía la sintaxis de JavaScript, por lo que un código escrito en JavaScript debería correr en TypeScript.

TypeScript está pensado para grandes proyectos que son compilados por el compilador de TypeScript y se traducen a código JavaScript original.

El compilador de TypeScript está escrito en TypeScript y compilado en JavaScript.

Su primera versión (0.8) fue lanzada en octubre de 2012, luego de 2 años de desarrollo. La versión 1.0 de TypeScript fue publicada en 2014.

La versión 2.0 fue lanzada el 22 de septiembre de 2016.

## Tipos de datos

- String: Cadena de caracteres.
- Number: Número.
- Boolean: Tipo de dato lógico que representa verdadero o falso.
- Array: Tipo de dato estructurado que permita almacenar varios datos.
- Tuple: Como el Array, sólo que Tuple almacena un número fijo de datos escritos.
- Enum: Enumeración.
- Any: La variable puede ser de cualquier tipo.
- Void: La función no devolverá ningún valor.
- Never: Tipo de valores que nunca se producen.

## Declaración de datos:

- `var isDone: boolean = false;`
- `var height: number = 7;`
- `var name: string = "rich";`
- `var list: number[] = [1,2,3];`
- `var notSure: any = 4;`
- `function warnUser(): void {  
 alert("Mensaje de alerta");  
}`

## Arreglos

En TypeScript, los arreglos se pueden presentar de 2 formas:

1.- nombreVariable: tipoDato[]

```
let arrayNumber: number[] = [1, 2, 3];
```

```
let arrayString: string[] = ['1', '2', '3'];
```

2.- nombreVariable: Array<tipoDato>

```
let arrayNumber: Array<number> = [1, 2, 3];
```

```
let arrayString: Array<string> = ['1', '2', '3'];
```

Se pueden combinar los tipos de datos, como un number y un string:

```
let arrayMixed: any[] = [2, '5', 3];
```

## Interfaces

Sirve para declarar objetos más complejos o estructurados:

```
// Declaramos las propiedades de la interface
```

```
interface Puppy {  
    name: string,  
    age: string  
};
```

```
// Declaración Válida
```

```
const puppy: Puppy = {  
    name: "Mascota",  
    age: 2  
};
```

```
// Declaración Inválida
```

```
const invalidPuppy: Puppy = {  
    eat: true  
};
```

```
// Esta declaración es inválida puesto que la prop eat no existe en la interface.
```

No es necesario llenar todas las propiedades de la interface:

// Interface

```
interface Puppy {  
    name: string,  
    age?: string  
};  
  
const puppy: Puppy = {  
    name: "Mascota"  
};
```

## Bibliografía:

- colaboradores de Wikipedia. (2020a, mayo 12). Biblioteca estándar de C. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Biblioteca\\_est%C3%A1ndar\\_de\\_C](https://es.wikipedia.org/wiki/Biblioteca_est%C3%A1ndar_de_C)
- colaboradores de Wikipedia. (2021, 25 febrero). TypeScript. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/TypeScript>
- Typed JavaScript at Any Scale. (s. f.). TypeScript. Recuperado 28 de febrero de 2021, de <https://www.typescriptlang.org/es/>