Operaciones E/S en C (II)

1. Introducción a las operaciones E/S
2. E/S carácter a carácter
3. E/S línea a línea
4. E/S con formato

1. Introducción a las operaciones E/S

- En C la E/S está orientada a caracteres, pero también pueden leerse o escribirse bytes (se conocecomo E/S a bajo nivel)
- Las principales funciones de E/S sobre la consola (entrada por teclado y salida por pantalla) están definidas en la biblioteca stdio.h
- Las funciones más utilizadas son:

  Entrada/Salida carácter a carácter:

  – getchar(), getc(), putchar(), putc()

  Entrada/Salida línea a línea:

  – gets(), puts()

  Entrada/salida con formato (ya vistas en el tema 1):

  – printf(), scanf()

2. E/S carácter a carácter

- **char getchar():** Lee un carácter de la entrada estándar (teclado).
- **putchar(char):** Escribe un carácter en la salida estándar (pantalla).

Ejemplo:

```
#include <stdio.h>
main() /* Cambia tipo de letra */ {
char ca;
  do { /*Lee carácter a carácter e intercambia mayúsculas y minúsculas */
      ca = getchar();
      if (islower(ca)) putchar (toupper(ca));
      else putchar (tolower(ca));
  } while (ca != '.'); /* Usa un punto para terminar */
  System("PAUSE");
  }
```

En el ejemplo han aparecido otras funciones de la biblioteca estándar:

- **islower(ca)**: Devuelve 1 si ca es una minúscula. 0 en caso contrario.
- **isupper(ca)**: Devuelve 1 si ca es una mayúscula. 0 en caso contrario.
- **toupper(ca)**: Convierte el carácter ca a mayúscula.
- **tolower(ca)**: Convierte el carácter ca a minúscula.

3. E/S línea a línea

- **gets():** Devuelve una cadena terminada con nulo en el array de caracteres que recibe como argumento. Al escribir una cadena y pulsar un retorno de carro (enter), gets() lee la cadena y a continuación le cambia el retorno de carro por el terminador nulo (*'\0').*
- **puts()**: Imprime en pantalla una cadena de caracteres, que pueden incluir los códigos que llevan asociados la barra invertida, como *'\n'.*

Ejemplo:

```
int getnum() {
char num[80], n;
  do {
     gets(num);
     if (! numero(num)) {
        puts ("Debe ser un numero\n");
        n = 0;
     }
     else n = 1;
  } while (!n);
  return (atoi(num));
}
```

En el ejemplo anterior **atoi()** convierte una cadena de caracteres ASCII a entero.

Otros ejemplos de funciones de la misma familia es **atof()**, ASCII a float.

4. E/S con formato

(YA VISTO EN TEMA 1)

Escritura:

int printf ("<cadena control>", <argumentos>);

- La cadena de control especifica el formato y el número de argumentos
- Los argumentos son las variables o expresiones a escribir
- Devuelve nº de argumentos correctamente escritos
- En la cadena de control pueden aparecer:
  - constantes carácter o cadena, que aparecen como tales,
  - constantes tipo carácter: \b, \n, \t, \', \",...
  - descriptores de formato, **%?**, que indican el formato con el que mostrarán los argumentos (equivalente al format de FORTRAN), donde **?** es uno de los siguientes:

Código formato	Descripción
%c	carácter sencillo
%d	entero
%e	real en notación científica
%f	real simple precisión en notación científica
%g	el más corto de %e, %f
%o	octal
%x	hexadecimal
%s	cadena de caracteres
%u	decimal sin signo

Los descriptores se pueden especificar mediante
%m.n?

Ejemplos:

%10d%10.5f %20s

Ejemplos:

```
printf (“Un entero en una linea: %d \n”, entero);
printf (“Dos enteros en una linea: %d, %d\n”, entero1, entero2);
printf (“Una cadena %s de caracteres.\n”, cadena);
printf (“Varios %d tipos %c mezclados %s\n”, entero, caracter, cadena);

for (i=1; i <= 100; i++) {
  printf (“%d \t”, i);
  if (i % 25 == 0) printf (“\n”);
}
```

Lectura:

int scanf (“<cadena de control>”, <argumentos>);

- cadena de control: idem que en el printf
- Devuelve número de argumentos leídos correctamente
- Los argumentos son las variables o expresiones a leer.
- **Los argumentos que sean de tipo dato-resultado o resultado y sean de tipos escalares, deben llevar delante el operador &**:
  &: indirección, pasa la dirección de la variable y no su valor.
  Esta es la forma en la que C modifica los valores de los argumentos de una función:

```
scanf (“%d”, &entero);              /* lee un entero */
scanf (“%d, %d”, &entero1, &entero2);  /* ha leido dos enteros*/
scanf (“%s”, cadena);  /* leida una cadena, que no es escalar*/
scanf (“%d %c %s\n”, &ent3, &c, string);
/* lee un entero, ent3, y un carácter, c, ambos escalares, y una cadena, string,
      que no es escalar*/
```