Cap. 9 - Entrada/Salida Formateada

Esquema

9.1 Introducción
9.2 Flujos
9.3 Formato de salida con impresión
9.4 Impresión de números enteros
9.5 Impresión de números de puntos flotantes
9.6 Impresión de cadenas y caracteres
9.7 Otros especificadores de conversión
9.8 Impresión con anchos de campo y precisiones
9.9 Uso de banderas en la cadena de control de formato de impresión
9.10 Impresión de literales y secuencias de escape
9.11 Formateo de la entrada con `scanf`

9.1 Introdución

- En este capítulo
  - Presentación de resultados
  - **`scanf`** y **`printf`**
  - Flujos (entrada y salidas)
    - **`gets`**, **`puts`**, **`getchar`**, **`putchar`** (**`<stdio.h>`**)

9.2 Flujos

- Flujos
  - Secuencia de caracteres organizados en líneas
    - Líneas de caracteres terminan con un carácter de nueva línea
    - ANSI C debe soportar líneas de hasta 254 caracteres
  - Realizan todas las entradas y salidas
  - Pueden a menudo ser redireccionados
    - Standard input - teclado
    - Standard output - monitor
    - Standard error - monitor

9.3 Salida formateado con `printf`

- **`printf`**
  - **`precisa de salida formateada`**
    - Especificaciones de conversión: flags, field widths, precisions, etc.
  - Puede realizar:
    - Redondeo de valores de punto flotante hasta un número indicado de posiciones decimales.
    - Alineación de una columna de número con puntos decimales que aparecen uno sobre el otro.
    - Justificación a la izquierda y justificación a la derecha de resultados.
    - Inserción de literales de carácter en la ubicación precisa de una línea de salida.
    - Representación de números de punto flotante en formato exponencial.
    - Representación de enteros sin signo en formato octal y decimal. Vea el apéndice E para mayor información respecto a los valores octales y hexadecimales.
    - Desplegado de todos los tipos de datos con anchos de campo y precisiones fijas.

9.3 Salida formateado con `printf` (II)

- Formato

  `printf( cadena de control de formato, otros argumentos );`

  – la cadena de control de formato describe el formato de salida

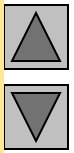
  – otros argumentos (los cuales son opcionales) corresponden a cada especificación de conversión de la cadena de control de formato.

    - Cada especificación de conversión comienza con un signo de porcentaje (`%`) que termina con un especificador de conversión.
    - Puede haber muchas especificaciones de conversión en una cadena de control de formato.

9.4 Imprimiendo Enteros

- Enteros
  - Número completo (sin punto decimal): `25`, `0`, `-9`
  - Positivo, negativo, o cero
  - Solo el signo menos se imprime por defecto

Especificador de conversión	Descripción
d	Despliega un entero decimal con signo.
i	Despliega un entero decimal con signo. [*Nota:* Los especificadores `i` y `d` son diferentes cuando se utilizan con `scanf`.]
o	Despliega un entero octal sin signo.
u	Despliega un entero decimal sin signo.
x o X	Despliega un entero hexadecimal sin signo. X provoca que se desplieguen los dígitos de 0 a 9 y las letras de A a F, y x provoca que se desplieguen los dígitos de 0 a 9 y las letras de a a f.
h o l (letra l)	Se coloca antes de cualquier especificador de conversión entera para indicar que se despliega un entero corto o largo, respectivamente. Las letras h y l son llamadas con más precisión *modificadores de longitud*.



Outline

1. Impresión

```
1   /* Fig 9.2: fig09_02.c */
2   /* Usando los especificadoes de conversion de enteros */
3   #include <stdio.h>
4
5   int main()
6   {
7      printf( "%d\n", 455 );
8      printf( "%i\n", 455 );  /* i igual a d en printf */
9      printf( "%d\n", +455 );
10     printf( "%d\n", -455 );
11     printf( "%hd\n", 32000 );
12     printf( "%ld\n", 2000000000 );
13     printf( "%o\n", 455 );
14     printf( "%u\n", 455 );
15     printf( "%u\n", -455 );
16     printf( "%x\n", 455 );
17     printf( "%X\n", 455 );
18
19     return 0;
20  }
```

Salida de Programa

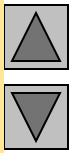
```
455
455
455
-455
32000
2000000000
707
455
65081
1c7
1C7
```

9.5 Imprimir Números de Punto Flotante

- Número de Punto Flotante
  - Tiene un punto decimal (**33.5**)
  - Notación Exponencial (la versión de la computadora de la notación científica)
    - **150.3** es **1.503 x 10²** en notación científica
    - **150.3** es **1.503E+02** en exponencial (**E** representa exponente)
    - use **e** o **E**
  - **f** - imprime un punto flotante con al menos un dígito a la izquierda del decimal
  - **g** (o **G**) - imprime en **f** o **e**(**E**) sin ceros de arrastre (**1.2300** se convierte **1.23**)
    - Use exponencial si el exponente es menos que **-4**, o mayor o igual a la precisión (6 dígitos por defecto)

9.5 Imprimir Números de Punto Flotante (II)

Especificador de conversión	Descripción
e o E	Despliega un valor de punto flotante con notación exponencial.
f	Despliega un valor de punto flotante con notación de punto fijo.
g o G	Despliega un valor de punto flotante con el formato de punto flotante f, o con el formato exponencial e (o E) basado en la magnitud del valor.
L	Se coloca antes del especificador de conversión para indicar que se desplegará un valor de punto flotante long double.



Outline

1. Impresión

```
1   /* Fig 9.4: fig09_04.c */
2   /* Imprimir números de punto flotante con
3      especificadores de conversión en punto flotante */
4
5   #include <stdio.h>
6
7   int main()
8   {
9      printf( "%e\n", 1234567.89 );
10     printf( "%e\n", +1234567.89 );
11     printf( "%e\n", -1234567.89 );
12     printf( "%E\n", 1234567.89 );
13     printf( "%f\n", 1234567.89 );
14     printf( "%g\n", 1234567.89 );
15     printf( "%G\n", 1234567.89 );
16
17     return 0;
18  }
```

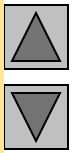
Salida

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

9.6 Imprimiendo Cadenas y Caracteres

- `c`
  - Imprime argument `char`
  - No puede ser usado para imprimir el primer carácter de una cadena
- `s`
  - Requiere de un puntero a `char` como un argumento
  - Imprime caracteres hasta que sea `NULL` (`'\0'`) encontrado
  - No puede imprimir un argumento `char`
- Recuerde
  - Simple comillas para carácter constante (`'z'`)
  - Doble comillas para cadenas `"z"` (el cual actualmente contiene dos caracteres: `'z'` y `'\0'`)

```c
/* Fig 9.5: fig09_05c */
/* Imprimir cadenas y caracteres */
#include <stdio.h>

int main()
{
   char character = 'A';
   char string[] = "This is a string";
   const char *stringPtr = "This is also a string";

   printf( "%c\n", character );
   printf( "%s\n", "This is a string" );
   printf( "%s\n", string );
   printf( "%s\n", stringPtr );

   return 0;
}
```



Outline

1. Inicializa variables

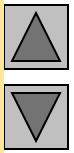
2. Imprime

Salidas

```
A
This is a string
This is a string
This is also a string
```

9.7 Otros especificadores de conversión

- **`p`**
  - Despliega un valor puntero (dirección)
- **`n`**
  - Almacena el número de caracteres ya emitidos por la declaración actual de `printf`
  - Toma un puntero a un entero como un argumento
  - El especificador de conversión %n no imprime valor alguno.
  - Cada llamada a **`printf`** retorna un valor
    - Número de caracteres de salida
    - Número negativo si ocurre un error
- **`%`**
  - Imprime un signo de porcentaje
  - `%%`



Outline

1. Inicializa variables

2. Imprime

```
1    /* Fig 9.7: fig09 07.c */
2    /* Uso de los especificadores de conversión p, n, y % */
3    #include <stdio.h>
4
5    int main()
6    {
7       int *ptr; /* define un apuntador a un int */
8       int x = 12345, y;
9
10      ptr = &x; /* asigna a ptr la dirección de x */
11      printf( "El valor de ptr es %p\n", ptr );
12      printf( "La direccion de x es %p\n\n", &x );
13
14      printf( "Total de caracteres impresos en esta linea:%n",&y );
15      printf( " %d\n\n", y );
16
17      y = printf( "Esta linea tiene 30 caracteres\n" );
18      printf( " se imprimieron %d caracteres\n\n", y );
19
20      printf( "Impresion de %% en una cadena de control de
21
22      return 0;
23   }
```

Salida

```
El valor de ptr es 0012FF78
La direccion de x es 0012FF78

Total de caracteres impresos en esta linea: 43

Esta linea tiene 30 caracteres
se imprimieron 31 caracteres

Impresion de % en una cadena de control de formato
```

9.8 Impresión con ancho de campos y precisiones

- Anchura de campo
  - Tamaño del campo en el que se imprimen los datos
  - Si el ancho es mayor que dato, por defecto justifica a la derecha
    - Si el ancho del campo es muy pequeño, incrementa para ajustar el dato
    - El signo menos utiliza la posición de un carácter en el campo
  - Ancho entero insertado entre el % y el especificador de conversión
  - **`%4d`** - anchura de campo de 4

9.8 Impresión con ancho de campos y precisiones (II)

- Precisión
  - El significado varía según el tipo de datos
  - Enteros (defecto 1) - mínimo número de digitos a imprimir
    - Si los datos son demasiado pequeños, prefijados con ceros
  - Punto Flotante - número de dígitos que aparecen después del decimal (**`e`** y **`f`**)
    - Para **`g`** - número máximo de dígitos significativos
  - Cadena - número máximo de caracteres a escribir desde la cadena

9.8 Impresión con ancho de campos y precisiones (III)

- Formato
  - Precisión: use un punto (.) y luego un número de precisión después de `%` **`(%.3f)`**
  - Puede ser combinado con ancho de campos **`(%5.3f)`**
  - Puede usar una expresión entera para determinar el ancho del campo y precisión
    - Use **`*`**
    - Ancho de Campo Negativo - justificado izquierda
    - Ancho de Campo Positivo - justificado derecho
    - Precision debe ser positiva

```
printf( "%*.*f", 7, 2, 98.736 );
```


```c
/* Fig 9.9: fig09 09.c */
/* Uso de la precisión durante la impresión de enteros
   números de punto flotante, y cadenas */
#include <stdio.h>

int main()
{
   int i = 873;                    /* inicializa el entero int i
   double f = 123.94536;           /* inicializa el double f */
   char s[] = "Feliz Cumpleanios";/* inicializa el arreglo char

   printf( "Uso de la precision en enteros\n" );
   printf( "\t%.4d\n\t%.9d\n\n", i, i );
   printf( "Uso de la precision en numeros de punto flotante\n"
   printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );
   printf( "Uso de la precision en cadenas\n" );
   printf( "\t%.11s\n", s );

   return 0; /* indica terminacion exitosa */
}
```

Outline

1. Inicializar variables

2. Imprimir

Salida

```
Uso de la precision en enteros
          0873
          000000873
Uso de la precision en numeros de punto flotante
          123.945
          1.239e+002
          124
Uso de la precision en cadenas
          Feliz Cumpl
```

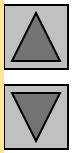
9.9 Uso de banderas en la cadena de control de formato de `printf`

- Banderas (Flags)
  - Capacidades de formato suplementarias
  - Coloca la bandera inmediatamente a la derecha del signo de porcentaje
  - Se pueden combinar varias banderas

Bandera	Descripción
– (signo menos)	Justifica la salida a la izquierda dentro del campo especificado.
+ (signo más)	Despliega el signo más que precede a los valores positivos, y un signo menos que precede a los valores negativos.
espacio	Imprime un espacio antes de un valor positivo no impreso con la bandera +.
#	Prefijo 0 para el valor de salida utilizado con el especificador de conversión octal o.
	Prefijo 0x o 0X para el valor de salida cuando se utiliza con el especificador de conversión de formato x o X.
	Fuerza la impresión del punto decimal de un número de punto flotante impreso con e, E, f, g o G que no contiene una parte fraccionaria. (Por lo general, el punto decimal solamente se imprime si le sigue un dígito.) Para los especificadores g y G, no se eliminan los ceros de acarreo.
0 (cero)	Rellena con ceros el principio de un campo.

```
1   /* Fig 9.11: fig09_11.c */
2   /* Justificación derecha e izquierda de valores */
3   #include <stdio.h>
4
5   int main()
6   {
7      printf( "%10s%10d%10c%10f\n\n", "hola", 7, 'a', 1.23 );
8      printf( "%-10s%-10d%-10c%-10f\n", "hola", 7, 'a', 1.23 );
9      return 0;
10  }
```

```
      hola         7         a  1.230000

hola      7         a         1.230000
```



Outline

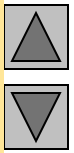
1. Impresion

Salida

```c
/* Fig 9.14: fig09 14.c */
/* Uso de la bandera # con los especificadores de conversión
   o, x, X y cualquier especificador de punto flotante */
#include <stdio.h>

int main()
{
   int c = 1427;
   double p = 1427.0;

   printf( "%#o\n", c );
   printf( "%#x\n", c );
   printf( "%#X\n", c );
   printf( "\n%g\n", p );
   printf( "%#g\n", p );

   return 0;
}
```

Salida

```
02623
0x593
0X593

1427
1427.00
```



Outline

1. Inicializa variables

2. Impresión

9.10 Impresión de literales y secuencias de escape

- Imprimiendo los literales
  - La mayoría de los caracteres pueden ser impresos
  - Existen caracteres “problemas”, tales como las comillas "
  - Deben ser representados por secuencias de escape
    - Representado por un **backslash** **** seguido por un carácter de escape

9.10 Impresión de literales y secuencias de escape (II)

Secuencia de escape	Descripción
\? (interrogación)	Despliega el carácter de signo de interrogación (?).
\\ (diagonal invertida)	Despliega el carácter de diagonal invertida (\).
\a (alerta o campana)	Provoca una alerta sonora (campana) o una alerta visual.
\b (retroceso)	Mueve el cursor una posición hacia atrás en la línea actual.
\f (nueva página o avance de página)	Mueve el cursor al inicio de la siguiente página lógica.
\n (nueva línea)	Mueve el cursor al principio de la siguiente línea.
\r (retorno de carro)	Mueve el cursor al principio de la línea actual.
\t (tabulador horizontal)	Mueve el cursor a la siguiente posición del tabulador horizontal.
\v (tabulador vertical)	Mueve el cursor a la siguiente posición del tabulador vertical.

9.11 Formato de entrada con `Scanf`

- **`scanf`**
  - Formato de entrada
  - Capacidades
    - Introduce todo tipo de datos.
    - Introduce caracteres específicos desde un flujo de entrada.
    - Ignora caracteres específicos del flujo de entrada.
- Formato

```
scanf(cadena de control de formato, otros argumentos);
```

  - La cadena de control de formato describe los formatos de la entrada
  - otros argumentos son apuntadores a las variables en las que se almacenará la entrada
  - puede incluir anchos de campo para leer un número específico de caracteres de la corriente

9.11 Formato de entrada con `Scanf` (II)

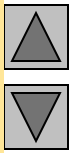
Especificador de conversión	Descripción
o	Lee un entero octal. El argumento correspondiente es un apuntador a un entero sin signo.
u	Lee un entero decimal sin signo. El argumento correspondiente es un apuntador a un entero sin signo.
x o X	Lee un entero hexadecimal. El argumento correspondiente es un apuntador a un entero sin signo.
h o l	Se coloca antes de cualquier especificador de conversión, para indicar que se introducirá un entero corto o largo, respectivamente.
Números de punto flotante	
e, E, f, g o G	Lee un valor de punto flotante. El argumento correspondiente es un apuntador a un valor de punto flotante.
l o L	Se coloca antes de cualquier especificador de conversión, para indicar que se introducirá un valor `double` o `long double`. El argumento correspondiente es un apuntador a una variable `double` o `long double`.
Cadenas y caracteres	
c	Lee un carácter. El argumento correspondiente es un apuntador a `char`; no agrega el carácter nulo (`'\0'`).
s	Lee una cadena. El argumento correspondiente es un apuntador a un arreglo de tipo `char` que sea lo suficientemente grande para almacenar la cadena y el carácter nulo (`'\0'`), el cual se agrega automáticamente.
Conjunto de exploración	
[caracteres de exploración]	Busca en una cadena un conjunto de caracteres almacenados en un arreglo.
Varios	
p	Lee una dirección de la misma forma que la dirección de salida con %p dentro de una instrucción `printf`.
n	Almacena el número de caracteres de entrada de `scanf`. El argumento correspondiente es un apuntador a un entero.
%	Ignora el signo de porcentaje en la entrada.

9.11 Formato de entrada con `Scanf` (III)

- Conjunto de exploración
  - Conjunto de caracteres entre corchetes **`[aeiou]`**
    - Precedido por el signo %
  - Escanea el flujo de entrada, buscando sólo los caracteres en el conjunto de escaneos
    - Cada vez que se produce una coincidencia, se almacena el carácter en el arreglo especificado
    - Detiene el escaneo una vez que se encuentra un desajuste
  - Conjunto de exploración invertidas
    - Use una tilde **`^`**: **`[^aeiou]`**
    - Condiciona que todos los caracteres que no se encuentran en el conjunto de exploración sean guardados

9.11 Formato de entrada con `Scanf` (IV)

- Saltarse los caracteres
  - Incluir el carácter a saltar en el control de formato
  - O, use ***** (asignación de carácter de supresión de la asignación)
    - Salta cualquier tipo de carácter sin almacenarlo



Outline

```
1   /* Fig 9.20: fig09 20.c */
2   /* Lectura de caracteres y cadenas */
3   #include <stdio.h>
4
5   int main()
6   {
7      char x, y[ 9 ];
8
9      printf( "Introduzca una cadena:" );
10     scanf( "%c%s", &x, y );
11
12     printf( "La entrada fue:\n" );
13     printf( "el caracter \"%c\" ", x );
14     printf( "y la cadena \"%s\"\n", y );
15
16     return 0; /* indica terminación exitosa */
17  }  /* indica terminación exitosa */
```

1. Inicializa variables

2. Entada

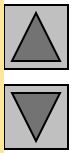
3. imprime

Salida

```
Introduzca una cadena: Domingo
La entrada fue:
el caracter "D" y la cadena "omingo"
```

```c
/* Fig 9.22: fig09_22.c */
/* Uso de un conjunto de exploración invertido */
#include <stdio.h>

int main()
{
   char z[ 9 ] = { '\0' };

   printf( "Introduzca una cadena: " );
   scanf( "%[^aeiou]", z );
   printf( "La entrada es \"%s\"\n", z );

   return 0;
}
```

```
Introduzca una cadena: Cadena
La entrada es "C"
```



Outline

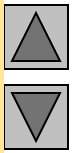
1. Incializa variables

2. Entradas

3. Imprime

Salida

```c
/* Fig 9.24: fig09 24.c */
/* Lectura y descarte de caracteres desde el flujo de entrada
#include <stdio.h>

int main()
{
   int month1, day1, year1, month2, day2, year2;

   printf( "Introduzca una fecha de la forma mm-dd-aaaa: " );
   scanf( "%d%*c%d%*c%d", &month1, &day1, &year1 );
   printf( "mes = %d  dia = %d  anio = %d\n\n",
      month1, day1, year1 );
   printf( "Introduzca una fecha de la forma mm/dd/aaaa: " );
   scanf( "%d%*c%d%*c%d", &month2, &day2, &year2 );
   printf( "mes = %d  dia = %d  anio = %d\n",
      month2, day2, year2 );

   return 0; /* indica terminación exitosa */
} /* fin de main */
```



Outline

1. Inicializa variables

2. Entrada

3. Impresión

Salida

```
Introduzca una fecha de la forma mm-dd-aaaa: 11-18-2003
mes = 11 dia = 18 anio = 2003

Introduzca una fecha de la forma mm/dd/aaaa: 11/18/2003
mes = 11 dia = 18 anio = 2003
```