Ampliación de E/S en C++

Manuel Montenegro
montenegro@fdi.ucm.es
Despacho B12

Manipuladores de salida

- Son objetos susceptibles de ser enviados a cualquier flujo de salida (en nuestro caso `cout`).
- No imprimen nada en pantalla por sí mismos, pero alteran la forma en la que se imprimen los siguientes elementos.

- Ejemplo:

```
cout << "Hola" << endl;
```

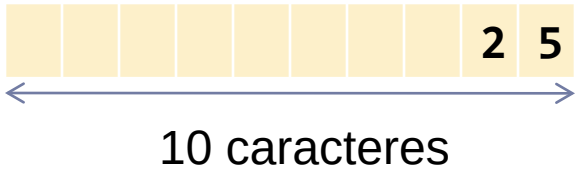
Imprime los siguientes elementos en una nueva línea

- Se encuentran definidos en el fichero `iomanip`

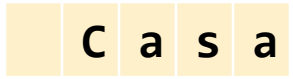
Manipuladores para justificación

- `setw(int n)`: Indica el número de caracteres que se utilizarán para representar el siguiente elemento.

```
cout << setw(10) << 25;
```




```
cout << setw(5) << "Casa";
```



Sólo afectan a la salida siguiente.

```
cout << setw(10) << 25 << endl << 32;
```

2 caracteres

Manipuladores para justificación

- Los manipuladores `left`, `right` e `internal`, combinados con `setw`, especifican como se alinea el texto con respecto a la anchura total.

```
cout << left << setw(6) << -25;
```
`-25   `

```
cout << right << setw(6) << -25;
```
`   -25`

```
cout << internal << setw(6) << -25;
```
`-   25`

Manipuladores para justificación

- Mediante el manipulador `setfill(char c)` puede indicarse el carácter de relleno (por defecto se utilizan espacios en blanco)

```
cout << setfill('*') << setw(10) << 102.3;
```

```
* * * * * 1 0 2 . 3
```

Otros manipuladores

- Para mostrar `float` y `double`
  - `setprecision(int n)`
  - `scientific`
  - `showpos`
- Para mostrar booleanos
  - `boolalpha / noboolalpha`

http://www.cplusplus.com/reference/iostream/manipulators/

Control de errores en entrada

- Supongamos que pedimos al usuario que introduzca un número.

```
int n;
cin >> n;
```

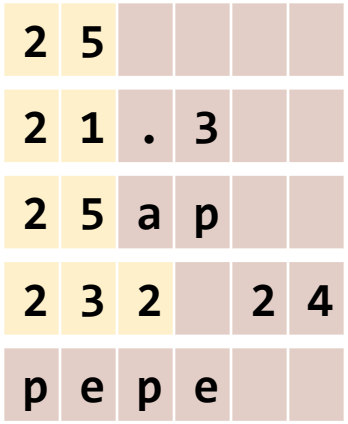
- ¿Qué ocurre si el usuario introduce…?
  - `25`
  - `21.3`
  - `25ap`
  - `232 24`
  - `pepe`

Control de errores en entrada

- La entrada no utilizada se guarda para la siguiente lectura.

```
int n;
cin >> n;
```

- 25
- 21.3
- 25ap
- 232 24
- pepe



Control de errores en la entrada

- El método `cin.good()` permite devuelve true si la última operación de lectura se realizó con éxito.

```
int n;
cin >> n;
if (!cin.good())
      cout << "Has introducido un valor incorrecto";
```

- En realidad, `cin` tiene una variable interna que describe si la última operación de entrada se ha realiza correctamente. `cin.good()` se limita a consultar esa variable.

Control de errores en la entrada

- ¿Qué se debe hacer en caso de fallo?
  - Debemos restablecer esa variable interna a su valor inicial (antes del fallo). De lo contrario, `cin.good()` devolverá siempre `false`, aunque las siguientes entradas fueran correctas.

    ```
    cin.clear();
    ```

  - Debemos descartar la salida incorrecta. De lo contrario se intentará leer la misma entrada cuando volvamos a leer de teclado.

    ```
    cin.ignore(500, '\n');
    ```

Ejemplo

```
int n;
cout << "Introduce un número: ";
cin >> n;

while (!cin.good()) {
   cout << "Eso no era un número." << endl;
   cout << "Introduce un número: ";
   cin.clear();
   cin.ignore(500,'\n');
   cin >> n;
}
```