



Universidad de Granada

[decsai.ugr.es](http://decsai.ugr.es)

# **Fundamentos de Programación**

**Francisco José Cortijo Bon**



**DECSAI**

**Departamento de Ciencias de la  
Computación e Inteligencia Artificial**



**Departamento de Ciencias de la  
Computación e Inteligencia Artificial**

**Universidad de Granada**

# **Fundamentos de Programación**

**Curso 2020-2021**

**Francisco José Cortijo Bon**

[cb@decsai.ugr.es](mailto:cb@decsai.ugr.es)

## **SEMINARIO: E/S en C++**

Última modificación: 20 de Septiembre de 2020

# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

- ▷ Un flujo proporciona un *canal de comunicación abstracto* entre:
  1. Memoria y dispositivos de E/S.
  2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

- ▷ Flujo: Una *secuencia* de bytes.

# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

- ▷ Un flujo proporciona un *canal de comunicación abstracto* entre:
  1. Memoria y dispositivos de E/S.
  2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

- ▷ Flujo: Una *secuencia* de bytes.

# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

▷ Un flujo proporciona un *canal de comunicación abstracto* entre:

1. Memoria y dispositivos de E/S.
2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

▷ Flujo: Una *secuencia* de bytes.

# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

▷ Un flujo proporciona un *canal de comunicación abstracto* entre:

- ➔ 1. Memoria y dispositivos de E/S.
- 2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

▷ Flujo: Una *secuencia* de bytes.



# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

▷ Un flujo proporciona un *canal de comunicación abstracto* entre:

1. Memoria y dispositivos de E/S.

➡ 2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

▷ Flujo: Una *secuencia* de bytes.

# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

▷ Un flujo proporciona un *canal de comunicación abstracto* entre:

1. Memoria y dispositivos de E/S.
2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

▷ Flujo: Una *secuencia* de bytes.



# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

- ▷ Un flujo proporciona un *canal de comunicación abstracto* entre:
  1. Memoria y dispositivos de E/S.
  2. Memoria y ficheros.

Proporciona una abstracción del problema de E/S que permite resolverlo sin considerar las particularidades de la fuente o el destino de los datos.

- ▷ Flujo: Una *secuencia* de bytes.

# Seminario: E/S en C++

C++ proporciona una serie de clases que se encargan de gestionar las operaciones de E/S. La gestión de E/S en C++ está basada en el concepto de *flujos* (del inglés, *streams*).

- ▷ Un flujo proporciona un *canal de comunicación abstracto* entre:
  1. Memoria y dispositivos de E/S.

**Flujo de texto:** *Secuencia de caracteres*

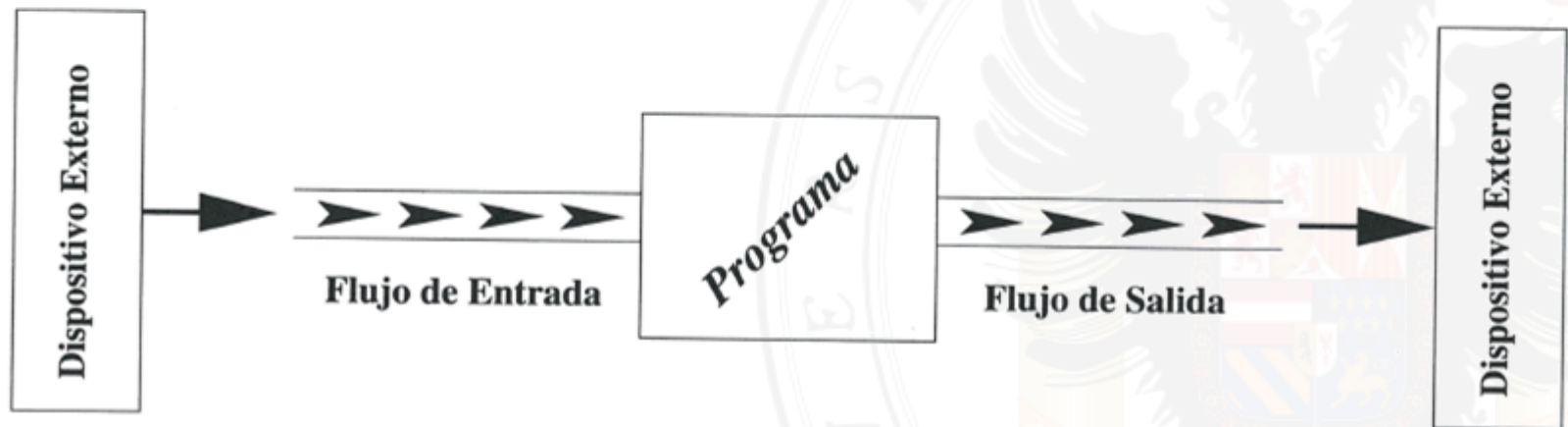
/S que permite re-  
a fuente o el des-

- ▷ **Flujo:** Una *secuencia de bytes*.

# Seminario: E/S en C++

- ▷ La aplicación es quien da significado al contenido (bytes) del flujo: éstos pueden representar caracteres, datos “binarios” (según su representación interna), imágenes, sonidos, video,...

*La E/S se produce en forma de flujos.*



# Seminario: E/S en C++

Los datos (sus valores) “circulan” ordenadamente, de uno en uno, a través de un flujo:

- desde el dispositivo de entrada a la memoria y
- desde la memoria al dispositivo de salida.

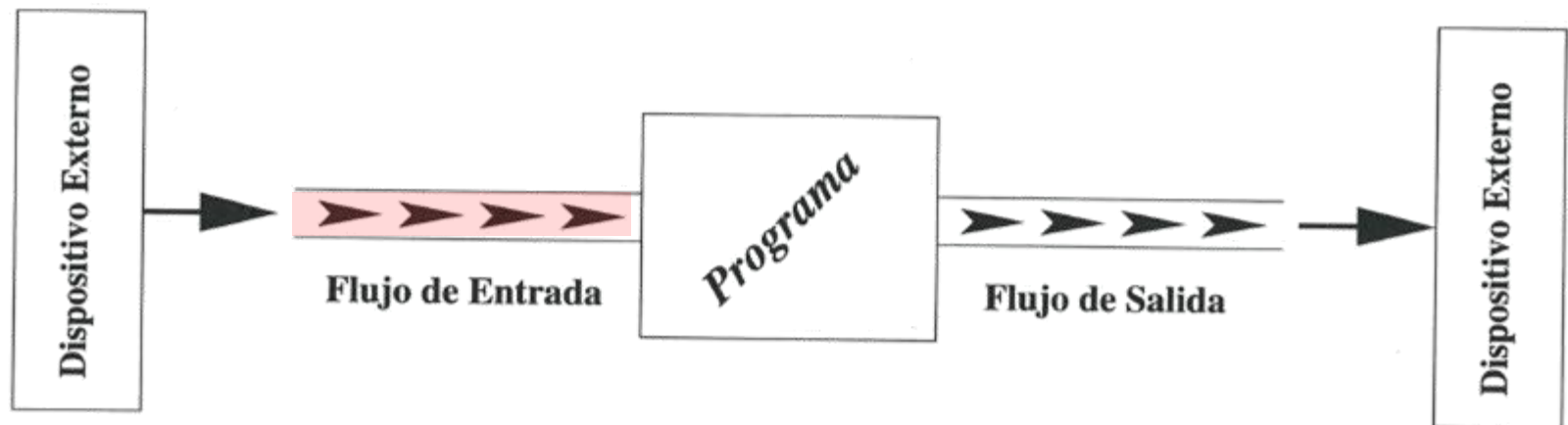
En esta asignatura en los flujos circulan datos en forma de **texto (caracteres)**:

- el dispositivo de entrada “inserta” caracteres, y
- el dispositivo de salida “acepta” caracteres.

# Seminario: E/S en C++

Los datos (sus valores) “circulan” ordenadamente, de uno en uno, a través de un flujo:

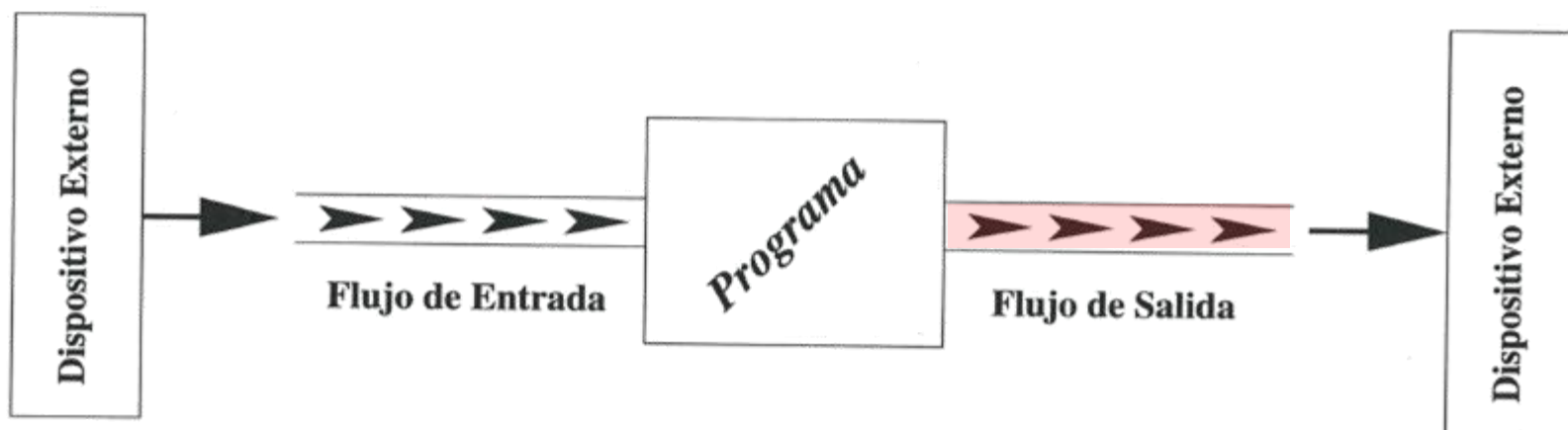
- ➔ desde el dispositivo de entrada a la memoria y
- desde la memoria al dispositivo de salida.



# Seminario: E/S en C++

Los datos (sus valores) “circulan” ordenadamente, de uno en uno, a través de un flujo:

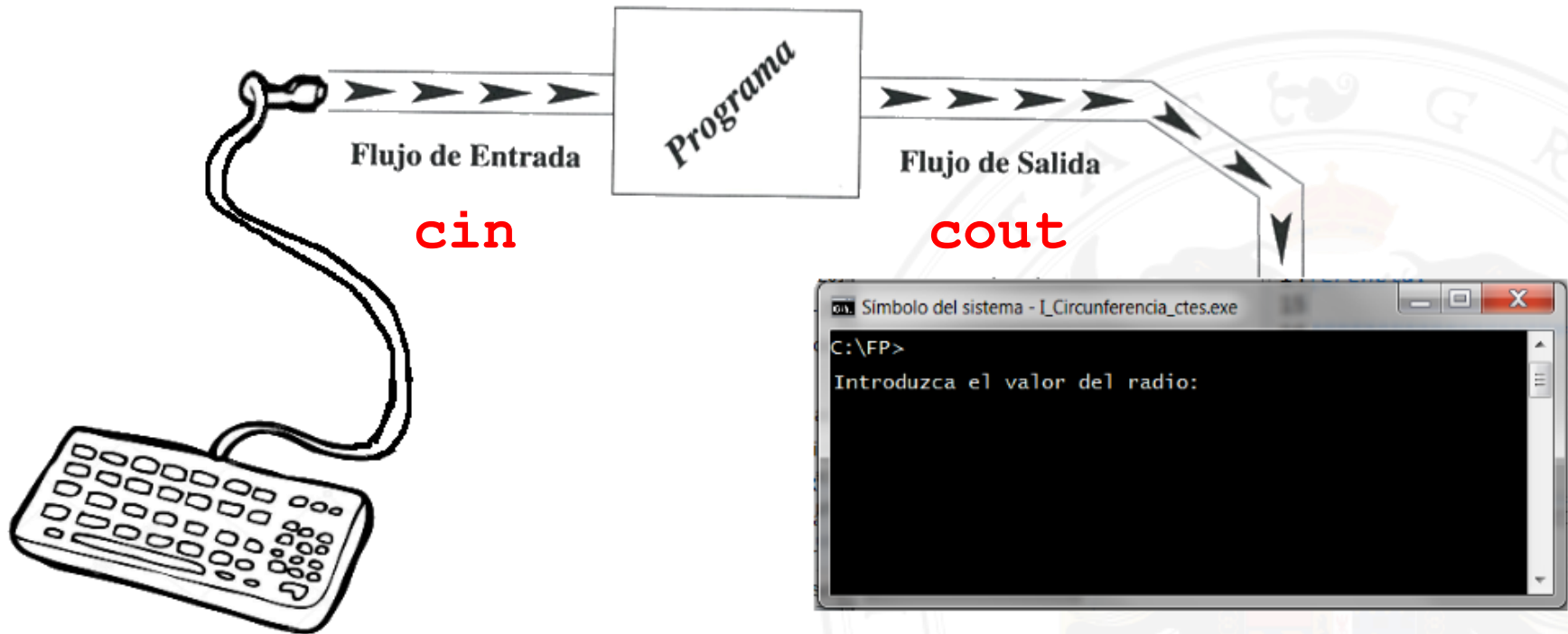
- desde el dispositivo de entrada a la memoria y
- desde la memoria al dispositivo de salida.





# Seminario: E/S en C++

Por defecto:



# Seminario: E/S en C++

Para realizar E/S en C++ hay que usar el paquete de recursos **iostream** (**i**input **o**utput **s**tream, *flujos de entrada/salida*):

```
#include <iostream>
```

Es conveniente (y cómodo) especificar que se usará el espacio de nombres **std**:

```
using namespace std;
```

# Seminario: E/S en C++

- Después de incluir `iostream` disponemos de los flujos:

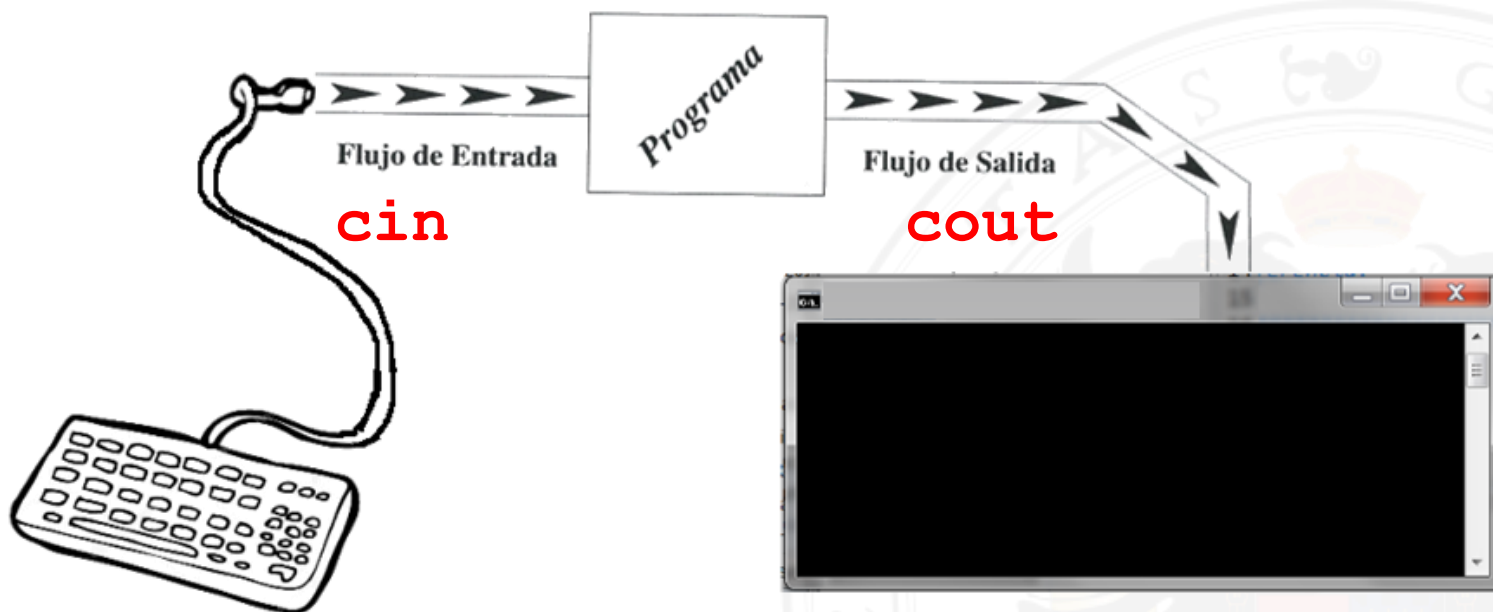
<code>cin</code>	→	console	<code>in</code>
<code>cout</code>	→	console	<code>out</code>

que están asociados (por defecto) al teclado (`cin`) y a la consola de salida de texto ó terminal (`cout`)

- Ambos flujos están desde entonces **operativos**. No es preciso hacer nada más.

# Seminario: E/S en C++

```
#include <iostream>
using namespace std;
```



# Seminario: E/S en C++

## El operador <<

**Inserta en un flujo de texto** (a la izquierda del operador) el contenido de lo que aparece a su derecha.

1. Si es un **literal de cadena de caracteres** se inserta “tal cual”.
2. Si es el **nombre de un dato** se inserta el valor textual (en forma de texto ó caracteres) del dato.

**El operador >> realiza la transformación a binario (el valor del dato) a texto.**

# Seminario: E/S en C++

## El operador <<

**Inserta en un flujo de texto** (a la izquierda del operador) el contenido de lo que aparece a su derecha.

1. Si es un **literal de cadena de caracteres** se inserta “tal cual”.
2. Si es el **nombre de un dato** se inserta el valor textual (en forma de texto ó caracteres) del dato.


El operador >> realiza la transformación a binario (el valor del dato) a texto.



# Seminario: E/S en C++

## El operador <<

**Inserta en un flujo de texto** (a la izquierda del operador) el contenido de lo que aparece a su derecha.


- 
1. Si es un **literal de cadena de caracteres** se inserta “tal cual”.
  2. Si es el **nombre de un dato** se inserta el valor textual (en forma de texto ó caracteres) del dato.

**El operador >> realiza la transformación a binario (el valor del dato) a texto.**

# Seminario: E/S en C++

## El operador <<

**Inserta en un flujo de texto** (a la izquierda del operador) el contenido de lo que aparece a su derecha.

1. Si es un **literal de cadena de caracteres** se inserta “tal cual”.
-  2. Si es el **nombre de un dato** se inserta el valor textual (en forma de texto ó caracteres) del dato.

**El operador >> realiza la transformación a binario (el valor del dato) a texto.**

# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hola, mundo";
7
8      return 0;
9  }
```

# Seminario: E/S en C++

## OPERADOR BINARIO << (Inserción en flujo)

Insertar en el flujo “de la izquierda” los caracteres “que están a la derecha”

```
1 #
2 u
3
4
5 {
6
7
8
9 }
```

```
int main()
```

```
{
```

```
    cout << "Hola, mundo";
```

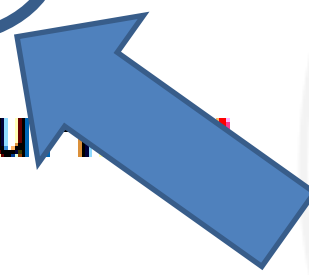
```
    return 0;
```

```
}
```

# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hola, mundo";
7
8      return 0;
9  }
```



Operando 1: **DÓNDE**

Indica el **FLUJO DE SALIDA**  
(CONSOLA O TERMINAL)

# Seminario: E/S en C++

## OPERANDO 2: QUÉ

Indica lo se va a insertar en el flujo: **literales** ó nombres de datos

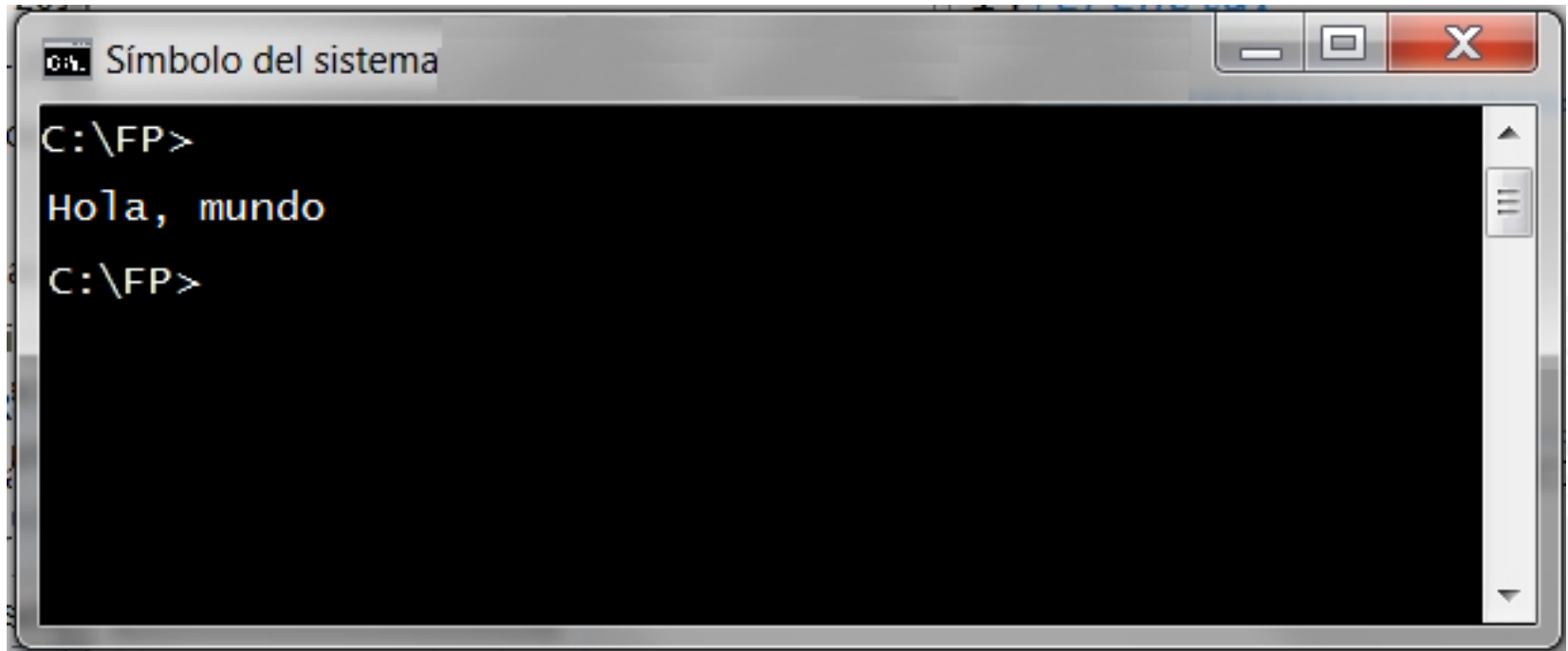
```

1  #include
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hola, mundo";
7
8      return 0;
9  }
```

"Hola, mundo";



# Seminario: E/S en C++



```

C:\FP>
Hola, mundo
C:\FP>
  
```

# Seminario: E/S en C++

## El operador >>

**Extrae de un flujo de texto** (a la izquierda del operador) los caracteres que forman ***una palabra***.


1. Convierte **la palabra** (caracteres) a un **valor** del tipo del dato que está a la derecha del operador.
2. Almacena el valor transformado en memoria, en la zona asociada al identificador del dato.

El operador >> realiza la transformación de texto (la palabra extraída) a binario (contenido del dato).

# Seminario: E/S en C++

## El operador >>

**Extrae de un flujo de texto** (a la izquierda del operador) los caracteres que forman ***una palabra***.


- 
1. Convierte **la palabra** (caracteres) a un **valor** del tipo del dato que está a la derecha del operador.
  2. Almacena el valor transformado en memoria, en la zona asociada al identificador del dato.

El operador >> realiza la transformación de texto (la palabra extraída) a binario (contenido del dato).

# Seminario: E/S en C++

## El operador >>

**Extrae de un flujo de texto** (a la izquierda del operador) los caracteres que forman ***una palabra***.

1. Convierte **la palabra** (caracteres) a un **valor** del tipo del dato que está a la derecha del operador.
-  2. Almacena el valor transformado en memoria, en la zona asociada al identificador del dato.

El operador >> realiza la transformación de texto (la palabra extraída) a binario (contenido del dato).

# Seminario: E/S en C++

## El operador >>

**Extrae de un flujo de texto** (a la izquierda del operador) los caracteres que forman *una palabra*.

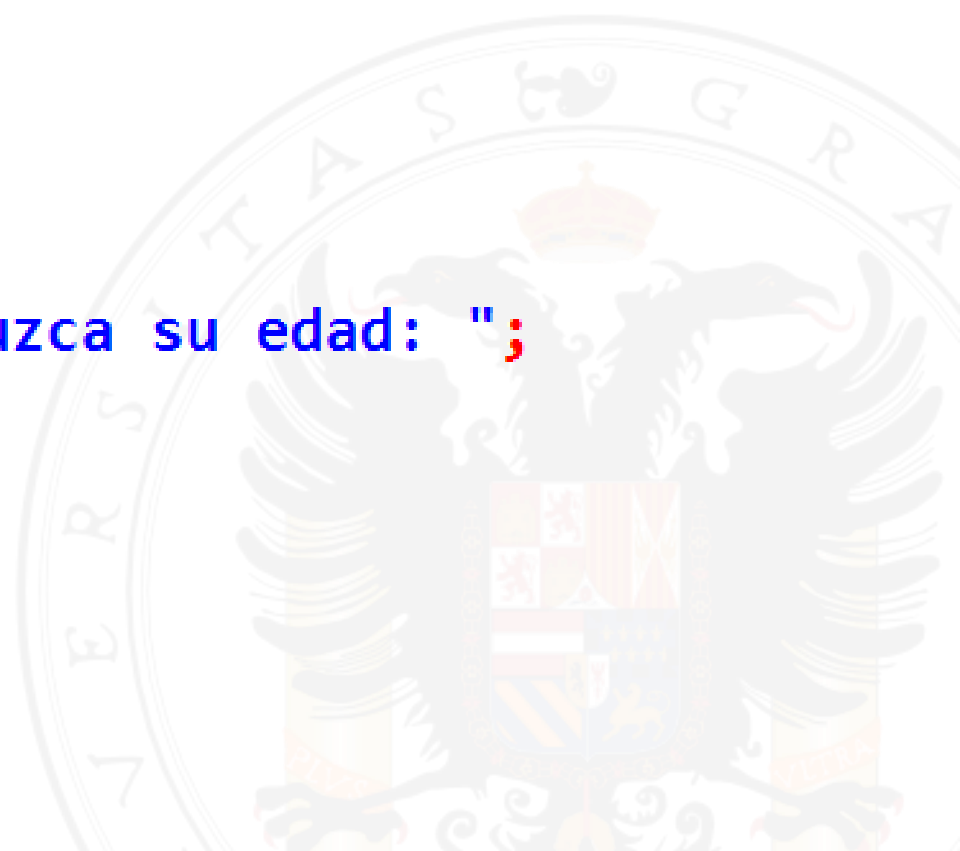
1. Convierte **la palabra** (caracteres) a un **valor** del tipo del dato que está a la derecha del operador.
2. Almacena el valor transformado en memoria, en la zona asociada al identificador del dato.

***El operador >> realiza la transformación de texto (la palabra extraída) a binario (contenido del dato).***

# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int edad;
7
8      cout << "Introduzca su edad: ";
9      cin >> edad;
10
11     //....
12
13     return 0;
14 }
```





# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int edad;
7
8      cout << "Introduce tu edad: ";
9      cin >> edad;
10
11     //....
12
13     return 0;
14 }
```

**OPERADOR BINARIO >>**  
(Extracción desde flujo)

Extrae del flujo “de la izquierda” una palabra, la transforma a un valor del tipo del dato “de la derecha” y lo asigna a éste.

# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int edad;
7
8      cout << "Introduzca su edad: ";
9      cin >> edad;
10     //...
11
12     return 0;
13 }
14

```

Operando 1: **DÓNDE**

Indica el **FLUJO DE ENTRADA** (TECLADO)

# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int edad;
7
8      cout << "Introdu
9      cin >> edad;
10
11     //....
12
13     return 0;
14 }
```

## OPERANDO 2: QUÉ

Indica el destino del valor leído y traducido.

La palabra leída se transforma a un valor `int` y se copia en `edad`

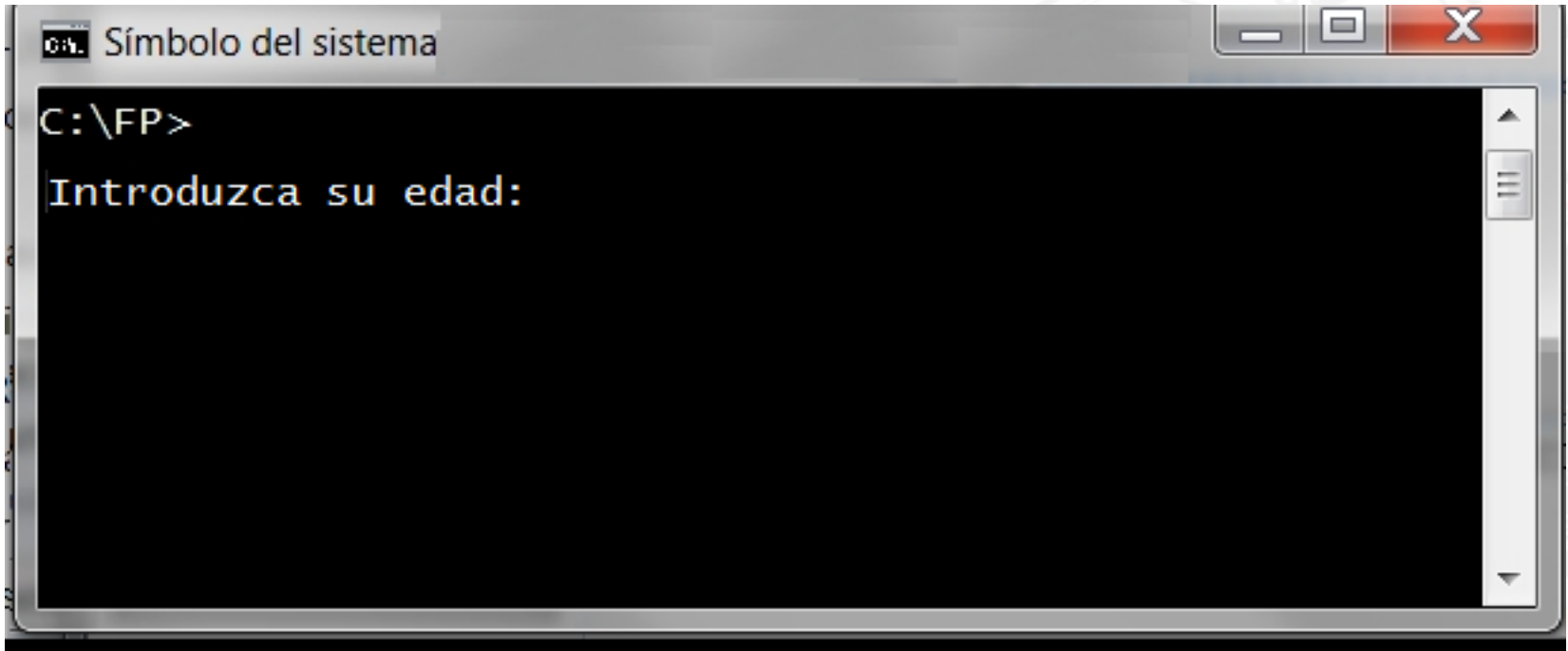
# Seminario: E/S en C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int edad;
7
8      cout << "Introduzca su edad: ";
9      cin >> edad;
10
11     //....
12
13     return 0;
14 }
```

# Seminario: E/S en C++

El programa detiene su ejecución (no hay ninguna palabra disponible para la variable `edad`) y espera a que el usuario introduzca *algo*:

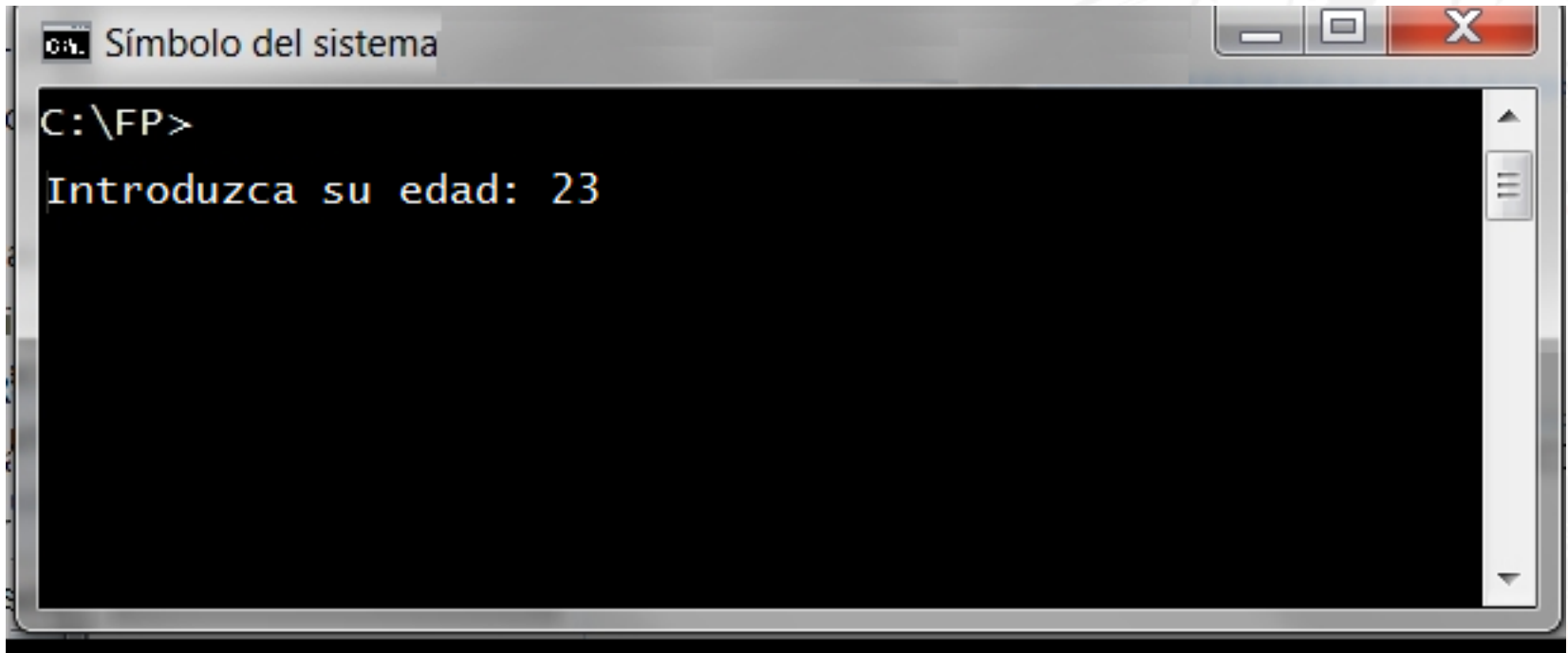


```

C:\FP>
Introduzca su edad:
  
```

# Seminario: E/S en C++

El usuario escribe una palabra (23) usando el teclado (cin está asociado al teclado) y al pulsar ENTER el programa continúa.

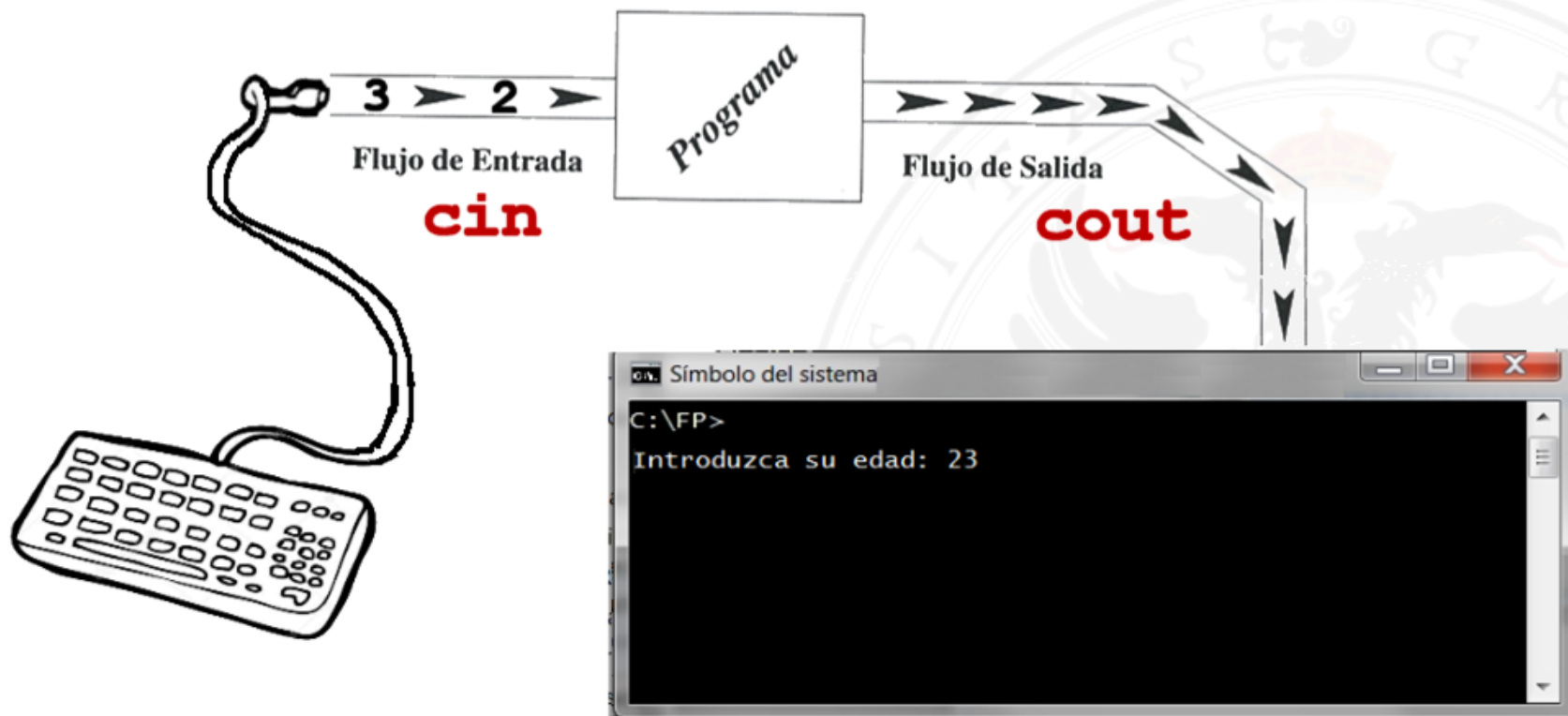


```

C:\FP>
Introduzca su edad: 23
  
```

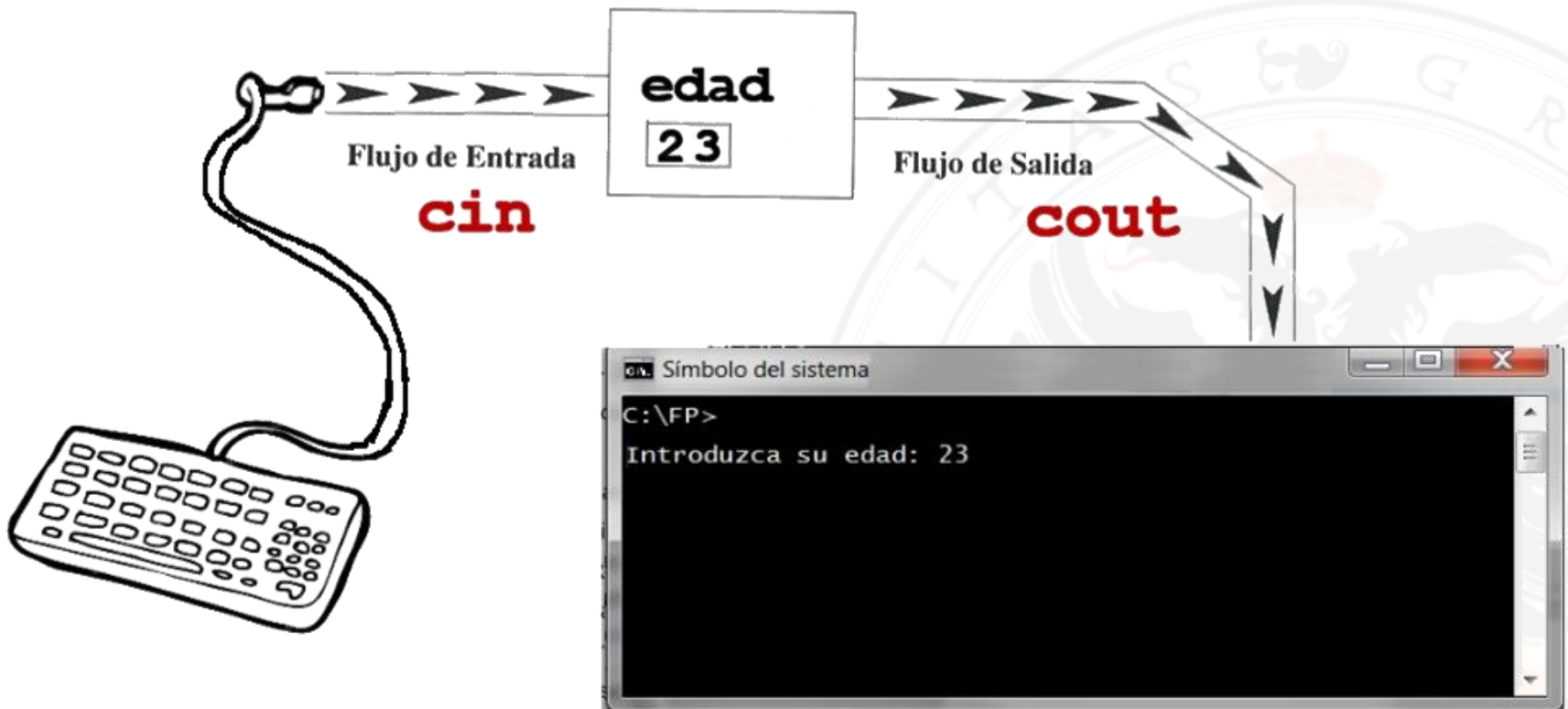
# Seminario: E/S en C++

El usuario escribe una palabra (23) usando el teclado (cin está asociado al teclado) y al pulsar ENTER el programa continúa.



# Seminario: E/S en C++

La palabra **23** se traduce a un valor **int** y se guarda en memoria (binario, representación interna 0...010111).





## Seminario: E/S en C++

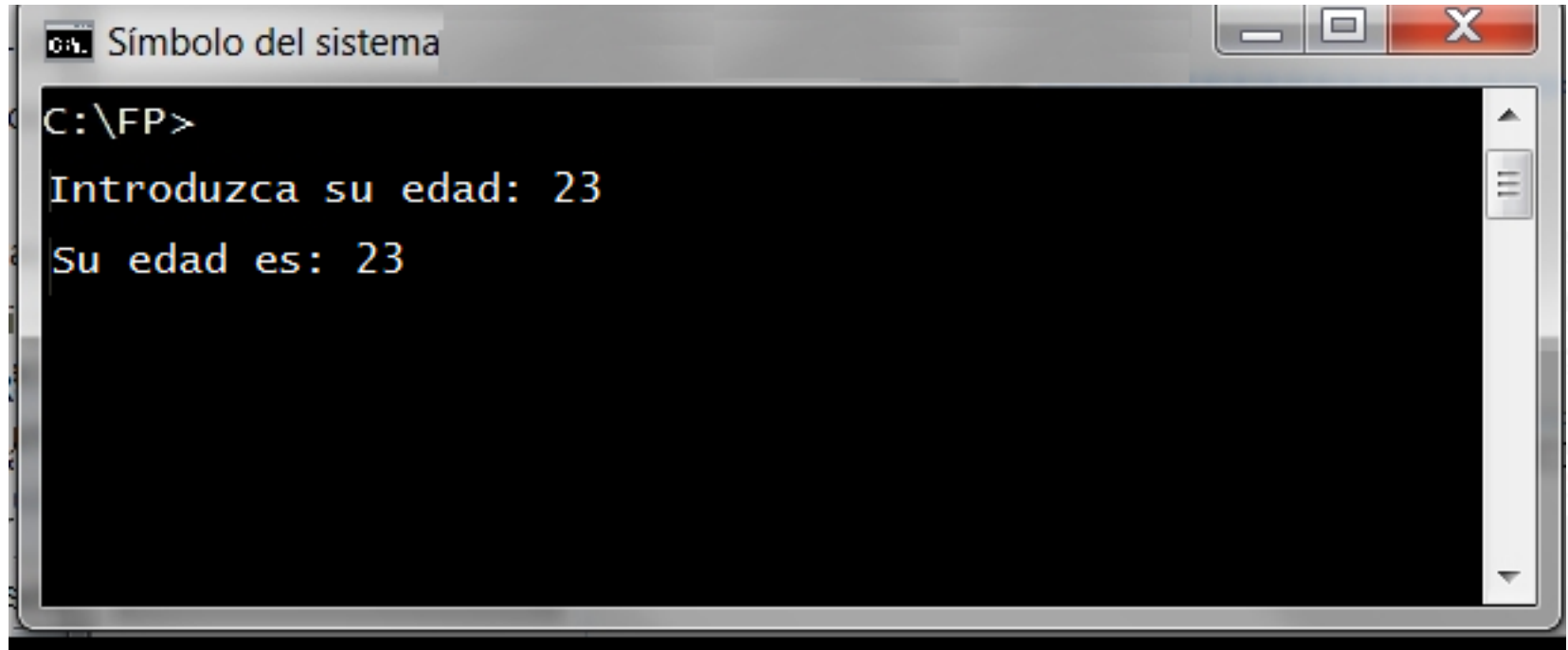
Si el programa mostrara el valor de `edad`:

```
cout << "Su edad es: " << edad;
```

Insertaría en `cout` los caracteres (en este orden):

1. `Su edad es:`  
(literal de cadena de caracteres)
2. `23`  
(valor *textual* de la variable `int` llamada `edad`)

# Seminario: E/S en C++



```

C:\FP>
Introduzca su edad: 23
Su edad es: 23
  
```

# Seminario: E/S en C++

## Personalizar las E/S (básico)

Para personalizar E/S en C++ hay que usar el paquete de recursos **iomanip** (**i**ntput **o**utput **m**anipulation):

```
#include <iomanip>
```

# Seminario: E/S en C++

## Ancho: `setw`

Indicamos el *número de casillas* que se van a emplear para mostrar un valor.

Si no se indica otra cosa, los “huecos” se rellenan con espacios.

## Relleno: `setfill`

Indicamos el *carácter* que se emplea para *rellenar* los “huecos”.

# Seminario: E/S en C++

Ancho: **setw**

Indicamos el *número de casillas* que se van a emplear para mostrar un valor.

Si no se indica otra cosa, los “huecos” se rellenan con espacios.

Relleno: **setfill**

Indicamos el *carácter* que se emplea para *rellenar* los “huecos”.

# Seminario: E/S en C++

```
#include <iostream>
#include <iomanip>
using namespace std;
```

...

```
int dato_una_cifra = 1;
int dato_dos_cifras = 23;
int dato_tres_cifras = 456;
```

# Seminario: E/S en C++

(Sin formato)

```
cout << dato_una_cifra << endl;  
cout << dato_dos_cifras << endl;  
cout << dato_tres_cifras << endl;
```

```
1  
23  
456
```

# Seminario: E/S en C++

(Con formato)

```
cout << setw(5) << dato_una_cifra << endl;  
cout << setw(5) << dato_dos_cifras << endl;  
cout << setw(5) << dato_tres_cifras << endl;
```

1
23
456



# Seminario: E/S en C++

## (Con formato y relleno)

```
cout << setfill('.') << setw(5) << dato_una_cifra << endl;  
cout << setfill('x') << setw(5) << dato_dos_cifras << endl;  
cout << setfill('*') << setw(5) << dato_tres_cifras << endl;
```

```
....1  
xxx23  
**456
```

# Seminario: E/S en C++

## Mostrar números reales

(Sin formato)

```
double dos_punto_cero = 2.0;  
double dos_punto_veintitres = 2.23;  
  
cout << dos_punto_cero << endl;  
cout << dos_punto_veintitres << endl;
```

```
2  
2.23
```

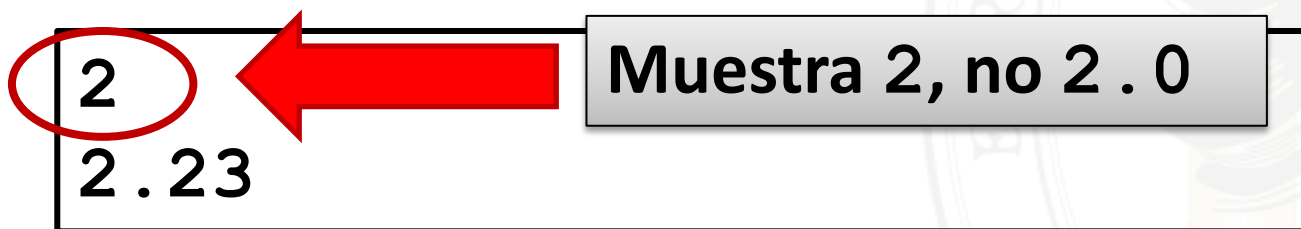
# Seminario: E/S en C++

## Mostrar números reales

(Sin formato)

```
double dos_punto_cero = 2.0;
double dos_punto_veintitres = 2.23;

cout << dos_punto_cero << endl;
cout << dos_punto_veintitres << endl;
```



# Seminario: E/S en C++

1) Para que SIEMPRE muestre el punto decimal:

```
cout.setf (ios::showpoint) ;
```

2) Para que muestre los números en notación de punto fijo (no científica):

```
cout.setf (ios::fixed) ;
```

3) Para indicar el número de cifras decimales:

```
setprecision (num.decimales)
```

# Seminario: E/S en C++

1) Para que SIEMPRE muestre el punto decimal:

```
cout.setf (ios::showpoint) ;
```

2) Para que muestre los números en notación de punto fijo (no científica):

```
cout.setf (ios::fixed) ;
```

3) Para indicar el número de cifras decimales:

```
setprecision (num.decimales)
```

# Seminario: E/S en C++

1) Para que SIEMPRE muestre el punto decimal:

```
cout.setf (ios::showpoint) ;
```

2) Para que muestre los números en notación de punto fijo (no científica):

```
cout.setf (ios::fixed) ;
```

3) Para indicar el número de cifras decimales:

```
setprecision (num.decimales)
```

# Seminario: E/S en C++

1) Para que SIEMPRE muestre el punto decimal:

```
cout.setf (ios::showpoint) ;
```

2) Para que muestre los números en notación de punto fijo (no científica):

```
cout.setf (ios::fixed) ;
```

3) Para indicar el número de cifras decimales:

```
setprecision (num.decimales)
```

# Seminario: E/S en C++

El valor que se indica en **setw** (casillas totales) es la **suma** de:

- Número de decimales (indicado con **setprecision**)
- 1 (por el punto decimal)
- El número de casillas para la parte entera. Si la parte entera no cupiera, se toman las que hagan falta.



# Seminario: E/S en C++

```
double dos_punto_cero = 2.0;  
double dos_punto_veintitres = 2.23;  
double dos_punto_trescientosveintiocho = 2.328;
```

```
cout.setf (ios::fixed) ;  
cout.setf (ios::showpoint) ;
```

```
cout << setw(6)<<setprecision(2)  
      << dos_punto_cero << endl;  
cout << setw(6)<<setprecision(2)  
      << dos_punto_veintitres << endl;  
cout << setw(6)<<setprecision(2)  
      << dos_punto_trescientosveintiocho << endl;
```

# Seminario: E/S en C++

```
double dos_punto_cero = 2.0;
double dos_punto_veintitres = 2.23;
double dos_punto_trescientosveintiocho = 2.328;
```

```
cout.setf (ios::fixed);
cout.setf (ios::showpoint);
```

```
cout << setw(6)<<setprecision(2)
      << dos_punto_cero << endl;
cout << setw(6)<<setprecision(2)
      << dos_punto_veintitres << endl;
cout << setw(6)<<setprecision(2)
      << dos_punto_trescientosveintiocho << endl;
```

```
2.00
2.23
2.33
```

# Seminario: E/S en C++

```
double dos_punto_cero = 2.0;
double dos_punto_veintitres = 2.23;
double dos_punto_trescientosveintiocho = 2.33;
```

```
cout.setf (ios::fixed);
cout.setf (ios::showpoint);
```

```
cout << setw(6)<<setprecision(2)
      << dos_punto_cero << endl;
cout << setw(6)<<setprecision(2)
      << dos_punto_veintitres << endl;
cout << setw(6)<<setprecision(2)
      << dos_punto_trescientosveintiocho << endl;
```

2 espacios+1dígito

1 casilla para el punto

2 dígitos

```

_____ 2.00
_____ 2.23
_____ 2.33
```

# Seminario: E/S en C++

```
double dos_punto_cero = 2.0;
double dos_punto_veintitres = 2.23;
double dos_punto_trescientosveintiocho = 2.33;
```

```
cout.setf (ios::fixed);
cout.setf (ios::showpoint);
```

```
cout << setw(6) << setprecision(2)
    << dos_punto_cero << endl;
cout << setw(6) << setprecision(2)
    << dos_punto_veintitres << endl;
cout << setw(6) << setprecision(2)
    << dos_punto_trescientosveintiocho << endl;
```

2 espacios+1dígito

1 casilla para el punto

2 dígitos

TOTAL = 6

```

_ 2.00
_ 2.23
_ 2.33
```

# Seminario: E/S en C++

```
double dos_punto_cero = 2.0;
double dos_punto_veintitres = 2.23;
double dos_punto_trescientosveintiocho = 2.328;
```

```
cout.setf (ios::fixed);
cout.setf (ios::showpoint);
```

```
cout << setw(7)<<setprecision(4)
      << dos_punto_cero << endl;
cout << setw(7)<<setprecision(4)
      << dos_punto_veintitres << endl;
cout << setw(7)<<setprecision(4)
      << dos_punto_trescientosveintiocho << endl;
```

```
2.0000
2.2300
2.3280
```

# Seminario: E/S en C++

```
double dos_punto_cero = 2.0;
double dos_punto_veintitres = 2.23;
double dos_punto_trescientosveintiocho = 2.328;
```

```
cout.setf (ios::fixed);
cout.setf (ios::showpoint);
```

```
cout << setw(7)<<setprecision(4)
      << dos_punto_cero << endl;
cout << setw(1)<<setprecision(4)
      << dos_punto_veintitres << endl;
cout << setw(3)<<setprecision(6)
      << dos_punto_trescientosveintiocho << endl;
```

```
2.0000
2.2300
2.328000
```