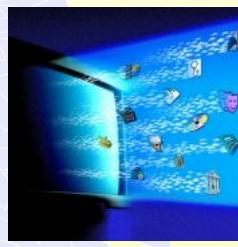


Metodología de la Programación

Tema 4.- Flujos (stream)

- Definiciones, tipos de flujos y operaciones**
- Operaciones de E/S con formato. Manipuladores**
- Operaciones de E/S sin formato**
- Control de estado y posicionamiento**
- Otras características de los flujos**



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 1

Definiciones, tipos de flujos y operaciones

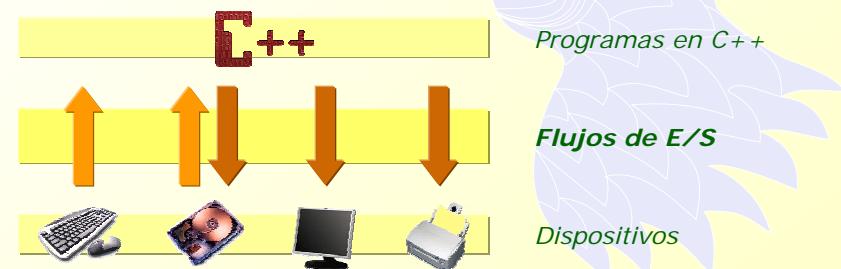
¿Qué es un flujo (stream)?

Es una **abstracción** que representa a un dispositivo y que permite realizar operaciones de E/S de datos con él (enviar o recibir datos).

Podemos ver un flujo como una secuencia de bytes que fluye desde o hacia algún dispositivo.

Un flujo siempre está **asociado** a un dispositivo sobre el que actuar.

Es frecuente que se trate de un dispositivo físico aunque podría ser otra cosa.



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 2

Definiciones, tipos de flujos y operaciones

¿Qué tipos de flujos hay?

Flujos de entrada (E/) : La secuencia de bytes se genera en el dispositivo y fluye *hacia el programa*.

Flujos de salida (/S) : La secuencia de bytes se genera en el programa y fluye *hacia el dispositivo*.

Flujos de entrada/salida (E/S) : La secuencia de bytes puede fluir en *ambos sentidos*.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 3

Definiciones, tipos de flujos y operaciones

Los flujos estándar

Ya están creados y definidos en el fichero de cabecera **iostream**
Cada flujo tiene asociado un dispositivo físico sobre el que actúa.

cin : Entrada estándar (teclado)	
cout : Salida estándar (monitor)	
cerr : Salida de errores estándar (monitor)	
clog : Salida de bitácora estándar (monitor)	

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 4

Definiciones, tipos de flujos y operaciones

Clases y ficheros de cabecera

Fichero de cabecera `istream`

Definición de la *clase istream* para gestionar flujos de entrada.

Todo lo que se define dentro de estos ficheros está incluido en el espacio de nombres (namespace) `std`.

Fichero de cabecera `ostream`

Definición de la *clase ostream* para gestionar flujos de salida.

Fichero de cabecera `iostream`

Incluye a `istream` y `ostream`.
 Definición de la *clase iostream* para gestionar flujos de entrada/salida.
 Declaración (y creación) de los *flujos por defecto*:
`cin` (de tipo `istream`).
`cout`, `cerr` y `clog` (de tipo `ostream`).

Cualquier cosa que se pueda hacer con un `ostream` (o un `istream`) también se puede hacer con un `iostream`.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 5

Definiciones, tipos de flujos y operaciones

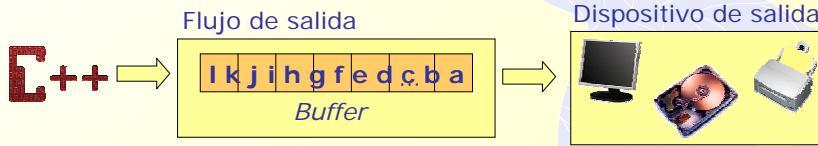
¿Cómo funciona un flujo?

Las operaciones de E/S con dispositivos suelen ser lentas (en comparación con la CPU o la transferencia a memoria).

Para hacer más eficiente la transferencia de datos: los bytes de los flujos no se envían de uno en uno.

Todo flujo mantiene en memoria un **buffer** (*memoria intermedia*).

Cuando el buffer está lleno se hace la transferencia a o desde el dispositivo.



El método `flush()` ordena la transferencia inmediata del contenido del buffer al dispositivo.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 6

Definiciones, tipos de flujos y operaciones

¿Cómo funciona un flujo?

Ejemplo: al hacer `cout` de algo no aparece en pantalla hasta que hemos enviado muchos datos y el buffer se ha llenado.

`endl` o `flush()` ordenan la transferencia del contenido del buffer.

```
int main() {
    cout << "Prueba";
    while (true);
}
```

```
int main() {
    cout << "Prueba" << endl;
    while (true);
}
```

```
int main() {
    cerr << "Prueba";
    while (true);
}
```

¿Qué aparece en pantalla?

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 7

Definiciones, tipos de flujos y operaciones

Tipos de operaciones

E/S con formato

Implica una **transformación** del dato que se transfiere.
Se transforma la representación interna a una representación comprensible por un usuario (en forma de secuencia de caracteres imprimibles).
Se controla el aspecto del dato que se transfiere.

Ejemplo: El número 65

Rep. interna, 2 bytes, binario natural = 0000 0000 0100 0001
Si lo mostramos en cout vemos “65” (lo hemos transformado)

E/S sin formato

La información se transfiere en bruto, sin transformaciones.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 8

<i>Definiciones, tipos de flujos y operaciones</i>		
Las operaciones más relevantes		
	ostream	istream
E/S con formato	<code>operator<<</code>	<code>operator>></code>
E/S sin formato	<code>put()</code> , <code>write()</code>	<code>get()</code> , <code>read()</code> , <code>peek()</code> <code>getline()</code> , <code>gcount()</code> , <code>readsome()</code> , <code>unget()</code> , <code>ignore()</code> , <code>putback()</code>
Estado del flujo	<code>bad()</code> , <code>clear()</code> , <code>eof()</code> , <code>fail()</code> , <code>good()</code> , <code>rdstate()</code> , <code>setstate()</code>	
Posicionamiento	<code>seekp()</code> , <code>tellp()</code>	<code>seekg()</code> , <code>tellg()</code>
Banderas de formato	<code>fill()</code> , <code>flags()</code> , <code>precision()</code> , <code>setf()</code> , <code>unsetf()</code> , <code>width()</code>	
Otras operaciones	<code>flush()</code>	
iostream		

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 9

Metodología de la Programación

Tema 9.- Flujos (stream)

Definiciones, tipos de flujos y operaciones

Operaciones de E/S con formato. Manipuladores

Operaciones de E/S sin formato

Control de estado y posicionamiento

Otras características de los flujos

Universidad
de Granada
DECSAI
Universidad de Granada

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 10

Operaciones de E/S con formato. Manipuladores

Salida con formato

operator<<

→ Recibe un *dato de entrada*, lo *transforma en caracteres imprimibles* y los *envía al ostream*.

→ Devuelve una referencia a un objeto *ostream* (el mismo al que hemos enviado los datos) → Permite encadenar salidas.

→ Está sobrecargado para los tipos estándar (int, float, double, ..., cadenas de C, string y punteros).

```
class ostream {
    ...
public:
    ostream & operator<<(const int &dato);
    ostream & operator<<(const float &dato);
    ostream & operator<<(const bool &dato);
    ...
};

ostream & operator<<(ostream & flujo, const string & dato);
...
```

En realidad, la definición es algo más compleja.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 11

Operaciones de E/S con formato. Manipuladores

Salida con formato

operator<< (ejemplo)

```
#include <iostream>
#include <string>

using namespace std;

int main(int argc, char *argv[])
{
    string cads="Hola";
    int x=74;
    double y=65.1234;
    int *ptr=&x;
    bool valor=false;
    cout << cads << endl;
    cout << x << " , " << y << endl;
    cout << ptr << " , " << valor << endl;
}
```

cout es un objeto de tipo ostream

Hola
74 , 65.1234
0x22ff44 , 0

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 12

Operaciones de E/S con formato. Manipuladores

Entrada con formato

operator>>

- Lee un dato (caracteres) del flujo, lo *transforma* al tipo del argumento y lo *asigna* al argumento (que es pasado por referencia).
- *Devuelve una referencia* a un objeto *istream* (el mismo del que estamos recibiendo los datos) → Permite encadenar entradas.
- Está sobrecargado para los tipos estándar (int, float, double, ..., cadenas de C, string y punteros).
- Los espacios en blanco, enter y TAB se consideran *separadores*.

```

class istream {
    ...
    public:
        istream & operator>>(int &dato);
        istream & operator>>(float &dato);
        istream & operator>>(bool &dato);
    ...
};

istream & operator>>(istream & flujo, string & dato);
...

```

En realidad, la definición es algo más compleja.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 13

Operaciones de E/S con formato. Manipuladores

Entrada con formato

operator>> (ejemplo)

```

#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    char cad[100];
    int x;
    float y;
    cin >> x >> y;
    cin >> cad;
    cout << x << endl << y << endl;
    cout << cad << endl;
}

```

cin es un objeto de tipo istream

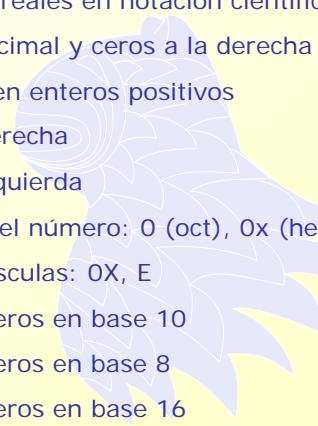
123	456.789	Hola Pepe
123	456.789	Hola

123.234	22	Hola Pepe
123	0.234	22

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 14

Operaciones de E/S con formato. Manipuladores

Banderas de formato



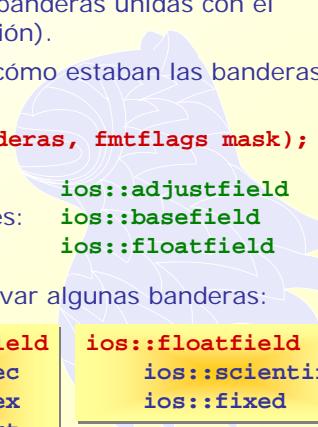
Los flujos mantienen una serie de **banderas (flags)** que permiten controlar la apariencia de los datos que leen o escriben:

ios::fixed	Escribir números reales en punto fijo
ios::scientific	Escribir números reales en notación científica
ios::showpoint	Mostrar punto decimal y ceros a la derecha
ios::showpos	Mostrar el signo en enteros positivos
ios::right	Alineación a la derecha
ios::left	Alineación a la izquierda
ios::showbase	Mostrar la base del número: 0 (oct), 0x (hex)
ios::uppercase	Mostrar en mayúsculas: 0X, E
ios::dec	Tratamos los enteros en base 10
ios::oct	Tratamos los enteros en base 8
ios::hex	Tratamos los enteros en base 16

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 15

Operaciones de E/S con formato. Manipuladores

Modificación del formato



setf() `fmtflags setf(fmtflags banderas);`

Para **activar** banderas del flujo.

banderas es un conjunto de una o más banderas unidas con el operador lógico a nivel de bit | (disyunción).

Devuelve el estado anterior al cambio (cómo estaban las banderas).

setf() `fmtflags setf(fmtflags banderas, fmtflags mask);`

mask puede ser alguno de estos valores:

ios::adjustfield	ios::basefield
ios::floatfield	

Esta segunda sintaxis se usa para activar algunas banderas:

ios::adjustfield ios::left ios::right ios::internal	ios::basefield ios::dec ios::hex ios::oct	ios::floatfield ios::scientific ios::fixed
---	---	---

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 16

Operaciones de E/S con formato. Manipuladores

Modificación del formato

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    cout.setf(ios::scientific,ios::floatfield);
    cout << 123.45 << endl;
    cout.setf(ios::fixed,ios::floatfield);
    cout << 123.45 << endl;
}
```

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    cout << 123 << endl;
    cout.setf(ios::showpos);
    cout << 123 << endl;
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 17

Operaciones de E/S con formato. Manipuladores

Modificación del formato

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    cout << 123 << endl;
    cout.setf(ios::hex,ios::basefield);
    cout << 123 << endl;
    cout.setf(ios::showbase);
    cout << 123 << endl;
    cout.setf(ios::oct,ios::basefield);
    cout << 123 << endl;
    cout.setf(ios::fmtflags(0),ios::showbase);
    cout << 123 << endl;
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 18

Operaciones de E/S con formato. Manipuladores

Modificación del formato

unsetf() `void unsetf(fmtflags banderas);`

Para **desactivar** banderas del flujo.

banderas es un conjunto de una o más banderas unidas con el operador lógico a nivel de bit | (disyunción).

flags() `fmtflags flags() const;`

Devuelve las banderas del flujo.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 19

Operaciones de E/S con formato. Manipuladores

Modificación del formato

precision()

```
unsigned int precision() const;
unsigned int precision(unsigned int p);
```

Para establecer (u obtener) la precisión con la que se muestran los números reales.

fill()

```
char fill() const;
char fill(char cf);
```

Para establecer (u obtener) el carácter de relleno cuando justificamos a izquierda o derecha (por defecto espacio en blanco).

width()

```
unsigned int width() const;
unsigned int width(unsigned int w);
```

Para establecer (u obtener) el número de posiciones que se van a ocupar al escribir un dato.

Sólo afecta a la siguiente operación de E/S.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Universidad de Granada) [navigation icons] 20

Operaciones de E/S con formato. Manipuladores

Modificación del formato

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    cout.width(20);
    cout.fill('.');
    cout.setf(ios::right,ios::adjustfield);
    cout << 123.45 << endl;
    cout << 123.45 << endl;
    cout.width(10);
    cout << 123.45 << endl;
}
```

.....123.45
123.45
....123.45

width sólo afecta a la siguiente operación de E/S

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 21

Operaciones de E/S con formato. Manipuladores

Manipuladores del flujo

Por comodidad existen una serie de funciones especiales (**manipuladores de formato**) que también permiten controlar el aspecto de la E/S.

Se definen en el fichero de cabecera **iomanip**

Se usan en mitad del envío/recepción de datos al flujo con los operadores **>>** y **<<**.

dec	showpos	setprecision(int n)
hex	noshowpos	setw (int n)
oct	showbase	setfill(int n)
fixed	noshowbase	setbase(int b)
scientific	uppercase	setiosflags(ios::fmtflags f)
left	nouppercase	resetiosflags(ios::fmtflags f)
right	showpoint	
boolalpha	noshowpoint	
noboolalpha	unitbuf	
	nounitbuf	

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 22

Operaciones de E/S con formato. Manipuladores

Manipuladores del flujo

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(int argc, char *argv[]) {
    cout << setbase(16) << showbase << 20 << endl;
    cout << hex << 20 << endl;
    cout << oct << noshowbase << 20 << endl;
    cout << dec << 0x20 << endl;
    cout << setprecision(3) << 2.123456 << endl;
    cout << setw(10) << 2.123456 << endl;
    cout << setw(10) << left << 2.123456 << endl;
    cout << setw(10) << left << setfill('*') << 2.123456 << endl;
    cout << setw(10) << right << setfill('*') << 2.123456 << endl;
}
```

 0x14 0x14 24 32 2.12	2.12 2.12***** *****2.12	
--	--------------------------------	--

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 23

Metodología de la Programación

Tema 9.- Flujos (stream)

- Definiciones, tipos de flujos y operaciones**
- Operaciones de E/S con formato. Manipuladores**
- Operaciones de E/S sin formato**
- Control de estado y posicionamiento**
- Otras características de los flujos**




ugr
 Universidad
 de Granada


DECSAI
 Universidad de Granada

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 24

Operaciones de E/S sin formato

Salida sin formato

put()

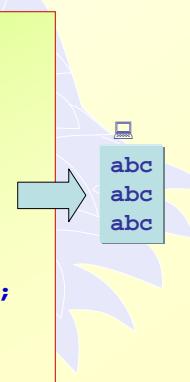
Miembro de *ostream*
`ostream & put(char c);`

Envía el carácter *c* al objeto *ostream* que lo invoca.
 Devuelve una referencia al flujo que lo invoca (**this*).

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]){
    char c1='a', c2='b';
    cout.put(c1);
    cout.put(c2);
    cout.put('c');
    cout.put('\n');

    cout.put(c1).put(c2).put('c').put('\n');

    cout << c1 << c2 << 'c' << '\n';
}
```



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 25

Operaciones de E/S sin formato

Salida sin formato

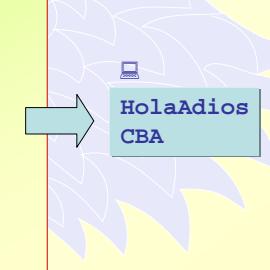
write()

Miembro de *ostream*
`ostream & write(const char *c, unsigned int n);`

Envía *n* bytes al objeto *ostream* que lo invoca.
c es un puntero al comienzo de los bytes que vamos a enviar.
 Devuelve una referencia al flujo que lo invoca (**this*).

No se suele usar con *cout*. Se suele usar para salida *sin formato* (en bruto) con *ficheros*.

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    char *cad="Hola";
    int x=0x414243;
    cout.write(cad,4);
    cout.write("Adios\n",6);
    cout.write((char*)&x,1);
    cout.write(((char*)&x)+1,2);
}
```



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 26

Operaciones de E/S sin formato

Entrada sin formato

Para leer un único carácter (byte):

get() Miembro de *istream*. (Sobrecargado)

```
int get();
istream& get(char& c);
```

Para leer varios caracteres (bytes):

get() Miembro de *istream*. (Sobrecargado)

```
istream& get(char* c, unsigned int n, char delim='\\n');
```

getline() Miembro de *istream*.

```
istream& getline(char* c, unsigned int n, char delim='\\n');
```

getline() Función no miembro

```
istream& getline(istream &f, string &c, char delim='\\n');
```

read() Miembro de *istream*.

```
istream& read(char *p, unsigned int n);
```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 27

Operaciones de E/S sin formato

Entrada sin formato

get() int get();

Lee un carácter (byte) y devuelve su valor.
Devuelve **EOF** (*End Of File*) si estamos al final del flujo.
EOF es una constante definida en *iostream* que suele tener el valor -1. Esta asociada a la combinación de teclas Ctrl+D.

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

int main(int argc, char *argv[]) {
    int ci;
    char cc;

    ci = cin.get();
    cout << ci << (char)ci << endl;

    cc = cin.get();
    cout << cc << (int)cc << endl;
}
```

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 28

Operaciones de E/S sin formato

Entrada sin formato

get() *istream& get(char &c);*

Lee un carácter (byte) y lo almacena en *c*.
Devuelve una referencia al *istream* que lo invocó (**this*).

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]) {
    char c1,c2,c3;
    cin.get(c1);
    cin.get(c2);
    cin.get(c3);
    cout << c1 << c2 << c3 << endl;
}
#include <iostream>
using namespace std;
int main(int argc, char *argv[]) {
    char c1,c2,c3;
    cin.get(c1).get(c2).get(c3);
    cout << c1 << c2 << c3 << endl;
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 29

Operaciones de E/S sin formato

Entrada sin formato

get() *istream& get(char* c, unsigned int n, char delim='\\n');*

Extrae bytes del flujo hasta que:

- Hemos leído *n-1* caracteres (bytes).
- O hemos encontrado el carácter *delim*.
- O hemos llegado al final del flujo o hay algún error de lectura.

Al finalizar la lectura:

- El carácter *delim* permanece en el flujo (no es extraído).
- Se añade un carácter '\0' al final de *c*.

Devuelve una referencia al *istream* que lo invocó (**this*).
c ha de tener memoria suficiente.

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]) {
    char c1[50],c2[50],c3[50];
    cin.get(c1,50);
    cin.get(c2,50,'a');
    cin.get(c3,50,'t');
    cout << c1 << endl;
    cout << c2 << endl;
    cout << c3 << endl;
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 30



Operaciones de E/S sin formato

Entrada sin formato

getline()

Miembro de *istream*.

istream& getline(char* c, unsigned int n, char delim='\\n');

La idea es casi la misma que la de *get()* salvo que *getline()* extrae *delim* del flujo (aunque tampoco lo almacena).

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

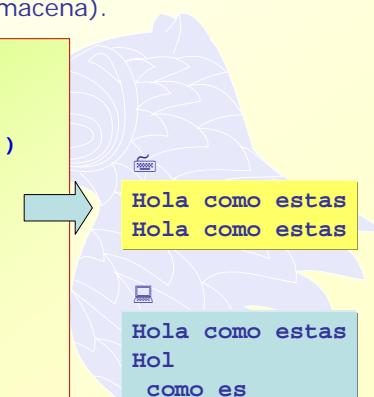
int main(int argc, char *argv[])
{
    char c1[50],c2[50],c3[50];
    cin.getline(c1,50);
    cin.getline(c2,50,'a');
    cin.getline(c3,50,'t');
    cout << c1 << endl;
    cout << c2 << endl;
    cout << c3 << endl;
}
```

Hola como estas

Hola como estas

Hola como estas

Hol
como es



The diagram illustrates the flow of data from the console input to the variable assignments in the code. It shows a blue arrow pointing from the console icon to the first assignment statement, another arrow from the console icon to the second assignment statement, and a third arrow from the console icon to the third assignment statement.



Operaciones de E/S sin formato

Entrada sin formato

getline()

Función **no** miembro.

istream& getline(istream &f, string &c, char delim='\\n');

Lee caracteres de un flujo y los almacena en un **string**.
Lee hasta encontrar el delimitador. Lo extrae pero no lo almacena.

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

int main(int argc, char *argv[])
{
    string cad1,cad2;
    getline(cin,cad1);
    cout << cad1 << endl;
    getline(cin,cad2,'m');
    cout << cad2 << endl;
}
```



Hola como estas
Hola como estas

Hola como estas
Hola co

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdивia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada)

Operaciones de E/S sin formato

Entrada sin formato

ignore()

Miembro de *istream*.

```
istream& ignore(unsigned int n=1, int delim=EOF);
```

Extrae caracteres del flujo y **no** los almacena en ningún sitio.

- Hasta que llegamos a *n* caracteres.
- O hasta que encontramos el carácter *delim*.
- *delim* también es extraído.

Devuelve una referencia al objeto *istream* que lo invocó (**this*).

```
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char *argv[])
{
    string c;
    cin.ignore(6);
    getline(cin,c);
    cout << c << endl;
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 33

Operaciones de E/S sin formato

Entrada sin formato

read()

Miembro de *istream*.

```
istream& read(char *p, unsigned int n);
```

Extrae un bloque de bytes consecutivos desde el flujo y los almacena en la memoria apuntada por *p*:

- Hasta que ha leído *n* bytes.
- O hasta que encuentra *EOF*.

Devuelve una referencia al objeto *istream* que lo invocó (**this*).

El flujo debe tener suficientes caracteres.

***p* ha de tener reservada suficiente memoria.**

readsome()

Miembro de *istream*.

```
unsigned int readsome(char *p, unsigned int n);
```

La función es la misma que la de *read()*. La diferencia es que devuelve el número de bytes que fueron leídos con éxito.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 34

Operaciones de E/S sin formato

Devolución de datos al flujo

putback()

Miembro de *istream*.

```
istream& putback(char c);
```

Devuelve al flujo de entrada el último carácter que fue leído mediante *get()*. *c* debe coincidir con ese carácter.

Devuelve una referencia al objeto *istream* que lo invocó (**this*).

unget()

Miembro de *istream*.

```
istream& unget();
```

Devuelve al flujo de entrada el último carácter que fue leído mediante *get()*.

Devuelve una referencia al objeto *istream* que lo invocó (**this*).

El número de caracteres consecutivos que pueden ser devueltos al flujo depende del compilador, el estándar sólo garantiza uno.

Operaciones de E/S sin formato

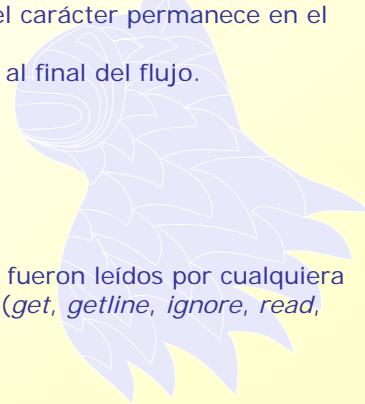
Devolución de datos al flujo

```
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char *argv[]) {
    unsigned char c;
    c = cin.get();
    cin.unget();
    if (isdigit(c)) {
        int n;
        cout << "Es un número" << endl;
        cin >> n;
        cout << n << endl;
    } else {
        string cad;
        cout << "Es una cadena" << endl;
        cin >> cad;
        cout << cad << endl;
    }
}
```

The diagram illustrates the execution of the provided C++ code. It shows two separate runs of the program. In the first run, the input '1234' is read by `cin.get()`, stored in variable *c*, and then immediately returned to the stream via `cin.unget()`. When `isdigit(c)` is checked, it returns true because '1234' is a digit. The program then prints "Es un número" followed by the value of *n* (which is 1234). In the second run, the input 'Hola' is read by `cin.get()`, stored in variable *c*, and then immediately returned to the stream via `cin.unget()`. When `isdigit(c)` is checked, it returns false because 'Hola' is not a digit. The program then prints "Es una cadena" followed by the string 'Hola'.

Operaciones de E/S sin formato

Consultas al flujo



peek()

Miembro de *istream*.

```
int peek();
```

Lee un carácter (byte) y devuelve su valor de forma similar a como lo hace *get()*. La diferencia es que el carácter permanece en el flujo para posteriores lecturas.

Devuelve EOF (*End Of File*) si estamos al final del flujo.

gcount()

Miembro de *istream*.

```
unsigned int gcount() const;
```

Devuelve el número de caracteres que fueron leídos por cualquiera de los métodos de lectura sin formato (*get*, *getline*, *ignore*, *read*, *readsome*).

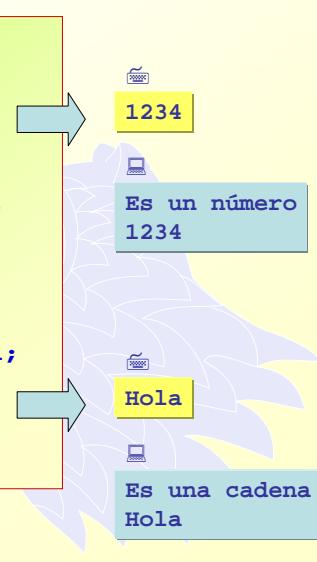
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 37

Operaciones de E/S sin formato

Consultas al flujo



```
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char *argv[]) {
    if (isdigit(cin.peek())) {
        int n;
        cout << "Es un número" << endl;
        cin >> n;
        cout << n << endl;
    } else {
        string cad;
        cout << "Es una cadena" << endl;
        cin >> cad;
        cout << cad << endl;
    }
}
```

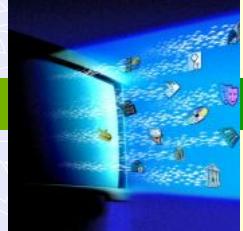


Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 38

Metodología de la Programación

Tema 9.- Flujos (stream)

- Definiciones, tipos de flujos y operaciones**
- Operaciones de E/S con formato. Manipuladores**
- Operaciones de E/S sin formato**
- Control de estado y posicionamiento**
- Otras características de los flujos**



UGR Universidad de Granada DECSAI Universidad de Granada

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 39

Control de estado y posicionamiento

Operaciones de consulta y modificación del estado

Los flujos mantienen un conjunto de **banderas de estado** que controlan la situación en la que se encuentra el flujo.

Hay un grupo de métodos que permiten **comprobar y cambiar** estas banderas de estado:

Consulta del estado:	eof() fail() good() bad() operator! rdstate()
Modificación del estado:	clear() setstate()



Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 40

Control de estado y posicionamiento

Operaciones de consulta y modificación del estado

eof()

```
bool eof() const;
```

Devuelve *true* si hemos alcanzado el final del flujo y *false* si no.

fail() **operator!()**

```
bool fail() const;
bool operator!() const { return fail(); };
```

fail() devuelve *true* si ha ocurrido algún error en la última operación sobre el flujo y *false* en caso contrario.

Si devuelve *true* la próxima operación que hagamos fallará.

clear()

```
void clear(iostate s=goodbit);
```

Modifica las banderas de estado del flujo.
Si se usa sin parámetros las "limpia".
Tras un estado de error (*fail*) es necesario hacer *clear()* para seguir trabajando.

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 41

Control de estado y posicionamiento

Operaciones de consulta y modificación del estado

Otros métodos:

bool good() const;
Devuelve *true* si la última operación sobre el flujo ha tenido éxito y *false* en caso contrario.

bool bad() const;
Devuelve *true* si el flujo está corrupto (ha habido errores irrecuperables) y *false* en caso contrario.

void setstate(iostate s);
Añade la bandera *s* a las banderas de estado del flujo.

iostate rdstate()
Devuelve las banderas de estado del flujo.

Banderas:
`ios::goodbit (0)`
`ios::eofbit`
`ios::failbit`
`ios::badbit`

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 42

Control de estado y posicionamiento

Operaciones de consulta y modificación del estado

Ejemplo: Leer caracteres desde teclado y escribirlos en pantalla (echo) hasta llegar a la marca EOF (Ctrl+D).

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

int main(int argc, char *argv[])
{
    int c;
    while (!cin.eof()) {
        c=cin.get();
        cout.put(c);
    }
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 43

Control de estado y posicionamiento

Operaciones de consulta y modificación del estado

Ejemplo: Lectura de tres datos enteros. Comprobamos si la lectura se ha podido hacer o no.

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

int main(int argc, char *argv[])
{
    int x;

    cin >> x;
    if (cin.fail()) cout << "Error" << endl;

    cin >> x;
    if (!cin) cout << "Error" << endl;

    cin >> x;
    if (!cin) cout << "Error" << endl;
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 44

Control de estado y posicionamiento

Operaciones de consulta y modificación del estado

Ejemplo: Lectura de tres datos enteros.

```
#include <iostream>
using namespace std;
void procesa_error() {
    cin.clear();
    cout << "Error" << endl;
    while (cin.get()!='\n');
}
int main(int argc, char *argv[]) {
    int x;

    cin >> x;
    if (cin.fail()) procesa_error();

    cin >> x;
    if (!cin) procesa_error();

    cin >> x;
    if (!cin) procesa_error();
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | Back | Forward | Home | 45

Control de estado y posicionamiento

Operaciones de posicionamiento

Podemos ver el flujo como una secuencia de bytes:

El flujo mantiene internamente un puntero a la posición en la que en cada momento toca leer o escribir.

Cada vez que leemos o escribimos un byte se avanza el puntero, de forma automática, al siguiente byte.

Existe un puntero de lectura y otro de escritura.

Ejemplo: Flujo de salida

```
flujo.put('b');
flujo.put('c');
flujo.put('d');
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) | Back | Forward | Home | 46

Control de estado y posicionamiento

Operaciones de posicionamiento

Podemos manipular la posición de los punteros con las funciones:

Puntero de lectura (*istream*):
`istream& seekg(int despl, int origen=ios::beg);`

Puntero de escritura (*ostream*):
`ostream& seekp(int despl, int origen=ios::beg);`

Desplazamos el puntero de forma relativa a *origen* (por defecto el comienzo del flujo) tantos bytes como indique *despl* (puede ser positivo o negativo).

Posibles valores para origen:

ios::beg	Relativo al comienzo del flujo (por defecto)
ios::cur	Relativo a la posición actual
ios::end	Relativo al final del flujo

int tellg() Devuelve la posición del puntero de lectura (*istream*)
int tellp() Devuelve la posición del puntero de escritura (*ostream*)

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 47

Metodología de la Programación

Tema 9.- Flujos (stream)

- Definiciones, tipos de flujos y operaciones**
- Operaciones de E/S con formato. Manipuladores**
- Operaciones de E/S sin formato**
- Control de estado y posicionamiento**
- Otras características de los flujos**



ugr Universidad de Granada

DECSAI
Universidad de Granada

Metodología de la Programación · Javier Martínez Baena/Joaquín Fdez-Valdivia · Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [Navigation icons] 48

Otras características de los flujos

Cuidado con la copia y construcción

No podemos definir un objeto de ostream o istream

Un flujo está asociado a un dispositivo ... ¿Qué significa esto?

```
main() {
    ostream x;
    ...
}
```

Los flujos no tienen definidos:

Ni constructor de copia
Ni `operator=`

Por tanto no es posible hacer asignaciones entre flujos.
Tampoco es posible crear nuevos flujos mediante constructor de copias.
En particular no podemos pasar un flujo como argumento por valor.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 49

Otras características de los flujos

Cuidado con la copia y construcción

Un flujo puede ser pasado como argumento de una función:
Por *referencia*.
Como un *puntero* al flujo.

NUNCA podrá ser pasado por valor.

~~void función(istream flujo)~~
~~{~~
~~...~~
~~}~~

```
void función(istream& flujo)
{
...
}
```

```
void función(istream* flujo)
{
...
}
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) [navigation icons] 50