Programación 1

Tema 14

Ficheros de texto

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

- **Textos** y **ficheros de texto**
- **Herramientas** de C++ para trabajar con ficheros de texto
- Resolución de problemas básicos con ficheros de texto
  - **Recorrido de un fichero** con información textual
  - **Creación de un fichero** con información textual

Texto

- Texto
  - Información estructurada mediante una secuencia de líneas (0, 1 o más líneas)
  - Cada línea está integrada por una secuencia de caracteres (0, 1 o más caracteres)
- Ejemplos
  - Teclado, pantalla, contenido de ficheros de texto

Texto

- Implementación
  - Secuencia de caracteres donde cada línea termina con uno o varios caracteres de control:
    - Linux
      - Carácter LF *(line feed)*, de código ASCII 10
    - Mac OS Classic (pre-Mac OS X)
      - Carácter CR *(carriage return)*, de código ASCII 13
    - Windows y algunos protocolos de internet
      - Caracteres CR+LF
  - En C++ representaremos ese carácter (o caracteres) como `'\n'` o `endl`, dependiendo del contexto.

Texto

Un soneto me manda hacer Violante
que en mi vida me he visto en tanto aprieto;
catorce versos dicen que es soneto;
burla burlando van los tres delante.

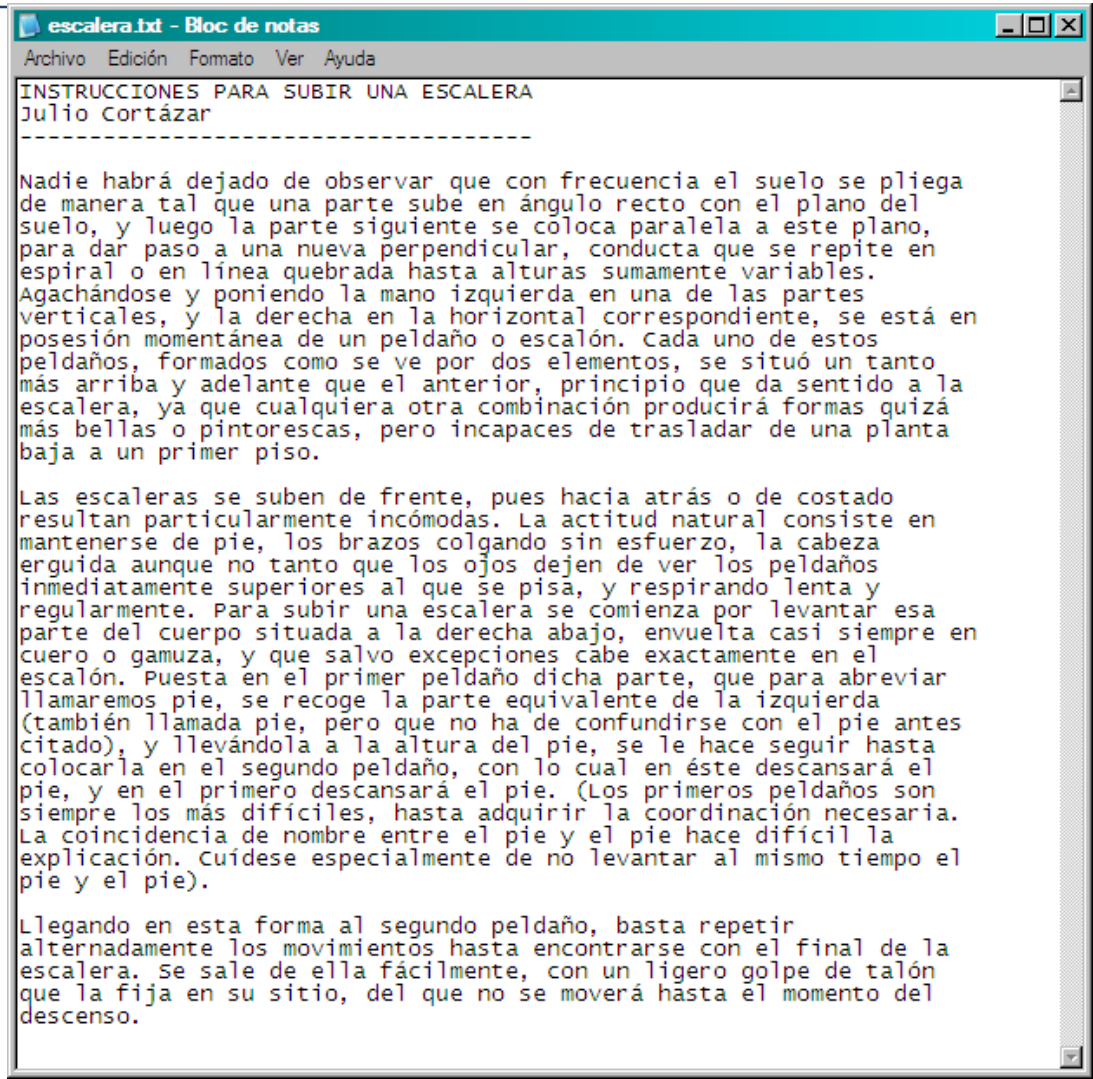
Yo pensé que no hallara consonante,
y estoy a la mitad de otro cuarteto;
mas si me veo en el primer terceto,
no hay cosa en los cuartetos que me espante.

Por el primer terceto voy entrando,
y parece que entré con pie derecho,
pues fin con este verso le voy dando.

Ya estoy en el segundo, y aun sospecho
que voy los trece versos acabando;
contad si son catorce, y está hecho.

Ficheros de texto

- Texto almacenado en un fichero
- Interpretación de la secuencia de *bytes* de un fichero como caracteres



Ficheros de texto

Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter	Código	Carácter
0	NUL	16	DLE	32		48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Ficheros de texto

```
01001001  01001110  01010011  01010100
01010010  01010101  01000011  01000011
01001001  01001111  01001110  01000101
01010011  00100000  01010000  01000001
01010010  01000001  00100000  01010011
01010101  01000010  01001001  01010010
00100000  01010101  01001110  01000001
00100000  01000101  01010011  01000011
01000001  01001100  01000101  01010010
01000001
```

Ficheros de texto

```
73   78   83   84   82
85   67   67   73   79
78   69   83   32   80
65   82   65   32   83
85   66   73   82   32
85   78   65   32   69
83   67   65   76   69
82   65
```

Ficheros de texto

```
I     N     S     T     R
U     C     C     I     O
N     E     S           P
A     R     A           S
U     B     I     R
U     N     A           E
S     C     A     L     E
R     A
```

Ficheros de texto

- Secuencias de *bytes* interpretadas como caracteres
- Estructurados en líneas

Ficheros de texto

- Secuencias de *bytes* interp
  caracteres
- Estructurados en líneas

pirata.txt - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
CANCIÓN DEL PIRATA
José de Espronceda
------------------

Con diez cañones por banda,
viento en popa, a toda vela,
no corta el mar, sino vuela
un velero bergantín.
Bajel pirata que llaman,
por su bravura, El Temido,
en todo mar conocido
del uno al otro confín.

La luna en el mar riela
en la lona gime el viento,
y alza en blando movimiento
olas de plata y azul;
y va el capitán pirata,
cantando alegre en la popa,
```

analisisTranvia.cc - Bloc de notas

Archivo Edición Formato Ver Ayuda

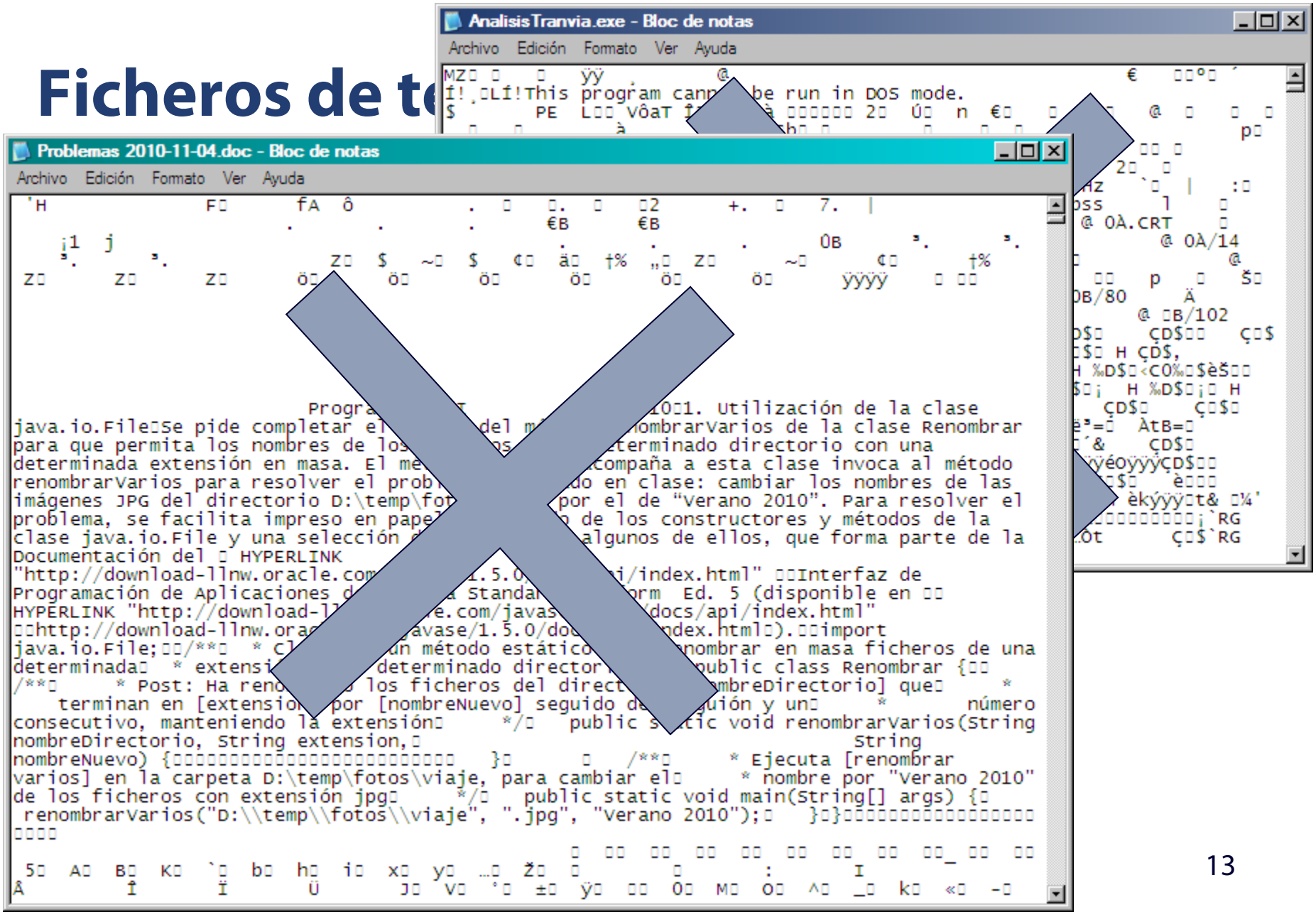
```
#include <iostream>
#include <iomanip>
#include <cstring>                          // funciones strlen(...) y strcmp(...)

#include "..\Tranvia\tranvia.h"             // módulo tranvía
#include "..\Operaciones\operaciones.h"     // módulo operaciones

using namespace std;

/*
 * Repertorio de códigos de órdenes válidas
 */
const char FIN[] = "FIN";
const char AYUDA[] = "AYUDA";
const char DATOS[] = "DATOS";
const char DIA[] = "DIA";
const char TOTAL[] = "TOTAL";
const char MIN[] = "MINIMO";
const char MAX[] = "MAXIMO";
const char VIAJEROS[] = "VIAJEROSDIA";
const char ACUMULADOS[] = "VIAJEROSACUMULADOS";
```

Ficheros de te



Problema 1

```
/*
 * Pre:  «nombreFichero» es el nombre de un fichero de
 *       texto existente y accesible para su lectura.
 * Post: Si el fichero de nombre «nombreFichero» se puede
 *       leer, asigna a «nLineas» el número de líneas que
 *       contiene del fichero, a «nCaracteres» el número
 *       de caracteres del mismo y a «lecturaOk», el valor
 *       «true». En caso contrario, asigna «false» a «lecturaOk»
 *       e informa del error ocurrido por «cerr».
 */
void contabilizar(const string nombreFichero,
                  unsigned &nLineas,
                  unsigned &nCaracteres,
                  bool &lecturaOk);
```

Una solución, leyendo carácter a carácter

```
void contabilizar(const string nombreFichero,
                  unsigned &nLineas, unsigned &nCaracteres, bool &lecturaOk){
    ifstream f;
    f.open(nombreFichero);
    if (f.is_open()) {
        nLineas = 0;
        nCaracteres = 0;
        char c;
        while (f.get(c)) {
            nCaracteres++;
            if (c == '\n') {
                nLineas++;
            }
        }
        f.close();
        lecturaOk = true;
    } else {
        cerr << "No se ha podido... \"" << nombreFichero << "\"." << endl;
        lecturaOk = false;
    }
}
```

Una solución, leyendo carácter a carácter

```
void contabilizar(const string nombreFichero,
          unsigned &nLineas, unsigned &nCaracteres,
          bool &lecturaOk) {
    ...
        nLineas = 0;
        nCaracteres = 0;
        char c;
        while (f.get(c)) {
            nCaracteres++;
            if (c == '\n') {
                nLineas++;
            }
        }
    ...
}
```

Una solución, leyendo línea a línea

```
void contabilizar(const string nombreFichero,
                  unsigned &nLineas, unsigned &nCaracteres, bool &lecturaOk){
    ifstream f;
    f.open (nombreFichero);
    if (f.is_open()) {
        nLineas = 0;
        nCaracteres = 0;
        string linea;
        while (getline(f, linea)) {
            nLineas++;
            nCaracteres += linea.length() + 1;
        }
        f.close();
        lecturaOk = true;
    } else {
        cerr << "No se ha podido abrir el fichero \"" << nombreFichero
             << "\"." << endl;
        lecturaOk = false;
    }
}
```

Una solución, leyendo línea a línea

```
void contabilizar(const string nombreFichero,
    unsigned &nLineas, unsigned &nCaracteres,
    bool &lecturaOk) {
    ...
        nLineas = 0; nCaracteres = 0;
        string linea;
        while (getline(f, linea)) {
            nLineas++;
            nCaracteres += linea.length() + 1;
        }
    ...
}
```

Ficheros de NIF

```
struct Nif {
    int dni;
    char letra;
};

...

const unsigned MAX_NUM_NIF = 700000;
Nif vector[MAX_NUM_NIF];
unsigned n;
```

Ficheros de NIF

- Se desea dar persistencia a vectores de datos de tipo `Nif`:
  - Definición de la sintaxis de un fichero de texto que almacena NIF
  - Diseño del código de dos funciones
    - Una función lea los datos de los NIF y almacene en un vector aquellos que sean válidos (su letra se corresponde con su DNI)
    - Otra función que escriba en un fichero los NIF presentes en un vector

Ficheros de NIF

Sintaxis

```
<fichero-nif> ::= { <nif> }
<nif> ::= <dni> <separador> <letra> <fin-línea>
<dni> ::= literal-entero
<letra> ::= "A" | "B" | "C" | "D" | …
          | "X" | "Y" | "Z"
<separador> ::= "-"
<fin-línea> ::= "\n"
```

Ficheros de NIF
Ejemplo

```
23087654-R
23208481-D
...
82413711-L
82534538-G
```

Ficheros de NIF
Escritura

```
/*
 * Pre:  ---
 * Post: Crea un fichero de texto de nombre
 *       «nombreFichero» en el que almacena los NIF de
 *       las primeras «n» componentes de «T», a razón de
 *       un NIF por línea, separando el número de DNI de
 *       la letra mediante un guion. Si el fichero se
 *       puede crear, asigna «true» a «escrituraOk». En
 *       caso contrario, asigna «false» y escribe un
 *       mensaje de error por «cerr».
 */
void escribirFicheroNif(const string nombreFichero,
                        const Nif T[], const unsigned n,
                        bool &escrituraOk);
```

Ficheros de NIF
Escritura

```
void escribirFicheroNif(const string nombreFichero,
     const Nif T[], const unsigned n, bool &escrituraOk) {
    ofstream f;
    f.open(nombreFichero);
    if (f.is_open()) {
        for (unsigned i = 0; i < n; i++) {
            f << T[i].dni << "-" << T[i].letra << endl;
        }
        f.close();
        escrituraOk = true;
    } else {
        cerr << "No se ha podido escribir el fichero \""
             << nombreFichero << "\"." << endl;
        escrituraOk = false;
    }
}
```

Ficheros de NIF
Lectura

```
/*
 * Pre:  El contenido del fichero de nombre «nombreFichero» sigue la sintaxis
 *       de la regla <fichero-nif> y el número de NIF válidos almacenados en
 *       el fichero «nombreFichero» es menor o igual a la dimensión del
 *       vector «T».
 * Post: Asigna a «nDatos» el número de NIF válidos del fichero y almacena en
 *       las primeras «nDatos» componentes del vector «T» la información de
 *       los NIF válidos almacenados en el fichero. A «nErroneos» le asigna
 *       el número total de NIF del fichero no válidos. Si el fichero se
 *       puede abrir, asigna «true» a «lecturaOk». En caso contrario, asigna
 *       «false» y escribe un mensaje de error por «cerr».
 */
void leerFicheroNif(const string nombreFichero, Nif T[],
                    unsigned &nDatos, unsigned &nErroneos, bool &lecturaOk);
```

Ficheros de NIF
Lectura

```
void leerFicheroNif(const string nombreFichero, Nif T[],
                    unsigned &nDatos, unsigned &nErroneos, bool &lecturaOk) {
    ifstream f;
    f.open(nombreFichero);
    if (f.is_open()) {
        nDatos = 0;
        nErroneos = 0;
        char ignorarGuion;
        while (f >> T[nDatos].dni >> ignorarGuion >> T[nDatos].letra) {
            if (esValido(T[nDatos])) {
                nDatos++;
            } else {
                nErroneos++;
            }
        }
        f.close();
        lecturaOk = true;
    } else {
        cerr << "No se ha podido leer del fichero \"" << nombreFichero << "\""
             << endl;
        lecturaOk = false;
    }
}
```

Ficheros de NIF
Lectura

```
void leerFicheroNif(const string nombreFichero, Nif T[],
                    unsigned &nDatos, unsigned &nErroneos,
                    bool &lecturaOk) {
    ...
        nDatos = 0;
        nErroneos = 0;
        char ignorarGuion;
        while (f >> T[nDatos].dni >> ignorarGuion
                 >> T[nDatos].letra) {
            if (esValido(T[nDatos])) {
                nDatos++;
            } else {
                nErroneos++;
            }
        }
    ...
}
```

¿Cómo se puede estudiar este tema?

- Repasando estas transparencias
- Trabajando con el código de estas transparencias
  - https://github.com/prog1-eina/tema-14-ficheros-de-texto
- Leyendo
  - Capítulo 14 de los apuntes del profesor Martínez, adaptados al curso 2021-22
    - Disponibles en Moodle
  - Tutoriales de *Cplusplus.com* (2000–2017)
    - «Basic Input/Output»: http://www.cplusplus.com/doc/tutorial/basic_io/
    - «Input/output with files»: http://www.cplusplus.com/doc/tutorial/files/
    - En ambos casos se introducen y explican más conceptos de los que se van a ver en este curso
- Problemas de las próximas clases
- Práctica 6