Programación 1

Tema 15

Trabajo con ficheros binarios

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

- Ficheros binarios
  - Diferencia con ficheros de texto
- Herramientas de C++ para trabajar con ficheros binarios
- Problemas básicos con ficheros binarios
  - Creación
  - Lectura

Ficheros binarios

- Almacenan una secuencia de datos codificados en binario
  - Cada dato se almacena como un grupo consecutivo de *bytes*
  - Para cada dato, se utiliza la misma codificación binaria que en la memoria del computador

Ficheros binarios

- Ejemplo:
  - Un dato de tipo **`int`** se almacena en un fichero binario como 4 *bytes* consecutivos en los que el entero está codificado en binario en complemento a 2.

Ficheros binarios

- Ventajas
  - Reducción del tamaño de los ficheros
  - Se facilitan las operaciones de lectura y escritura
    - Simplificación de las instrucciones que es necesario programar
    - Reducción del tiempo de lectura o escritura
- Desventajas
  - No legibles por seres humanos
  - Pueden aparecer problemas de portabilidad

Diferencias entre un fichero binario y un fichero de texto

- Ejemplo: 26173, dato de tipo **`int`**
  - Codificación en un **fichero de texto**:
    - 00110010 00110110 00110001 00110111 00110011
    - (= Secuencia de *bytes* 50, 54, 49, 55 y 51)
    - (= Secuencia de caracteres de códigos 50, 54, 49, 55 y 51)
    - (= Secuencia de caracteres '2', '6', '1', '7' y '3')
  - Codificación en un **fichero binario**:
    - 00111101 01100110 00000000 00000000
    - (= Secuencia de *bytes* 61, 102, 0 y 0)
    - (= 4 *bytes* que codifican el número 26173 en base 2 en complemento a 2, con el byte menos significativo en primer lugar)

Diferencias entre un fichero binario y un fichero de texto

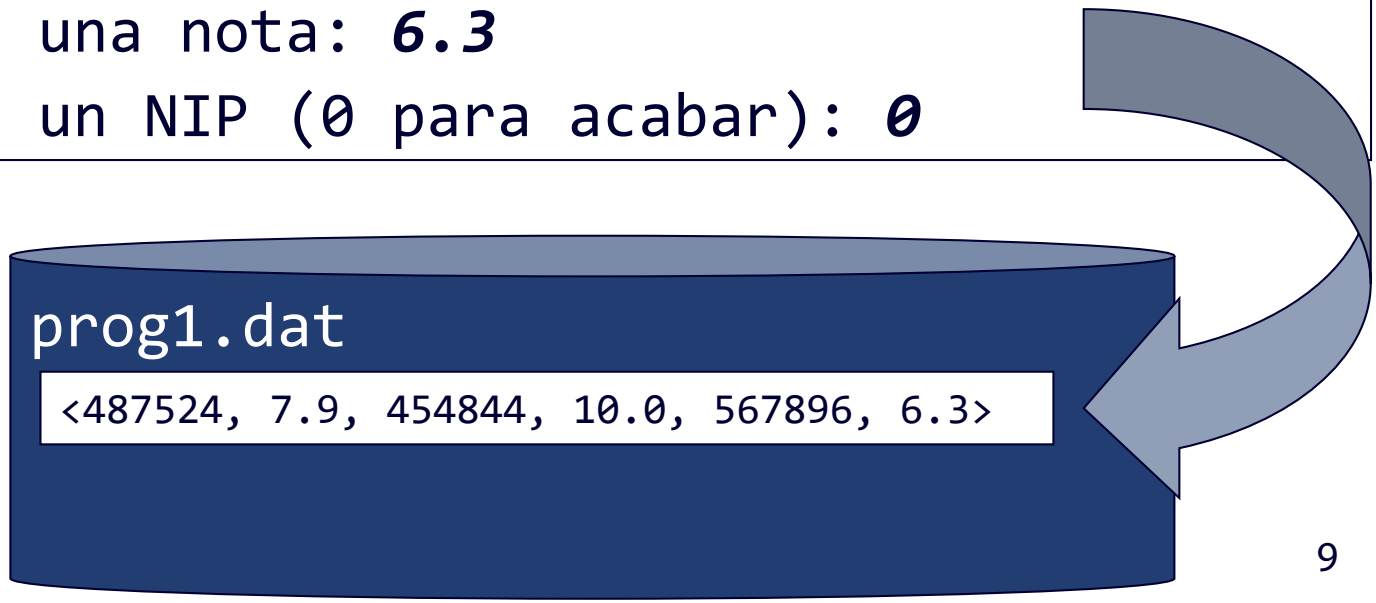
	Fichero de texto	Fichero binario
Interpretación de la secuencia de bytes	Caracteres	Codificación interna binaria de datos
¿Estructurado en líneas?	Sí	No
Necesidad de separadores entre datos	Habitualmente, sí	Habitualmente, no
Legible por una persona	Sí	No

Trabajo con ficheros binarios

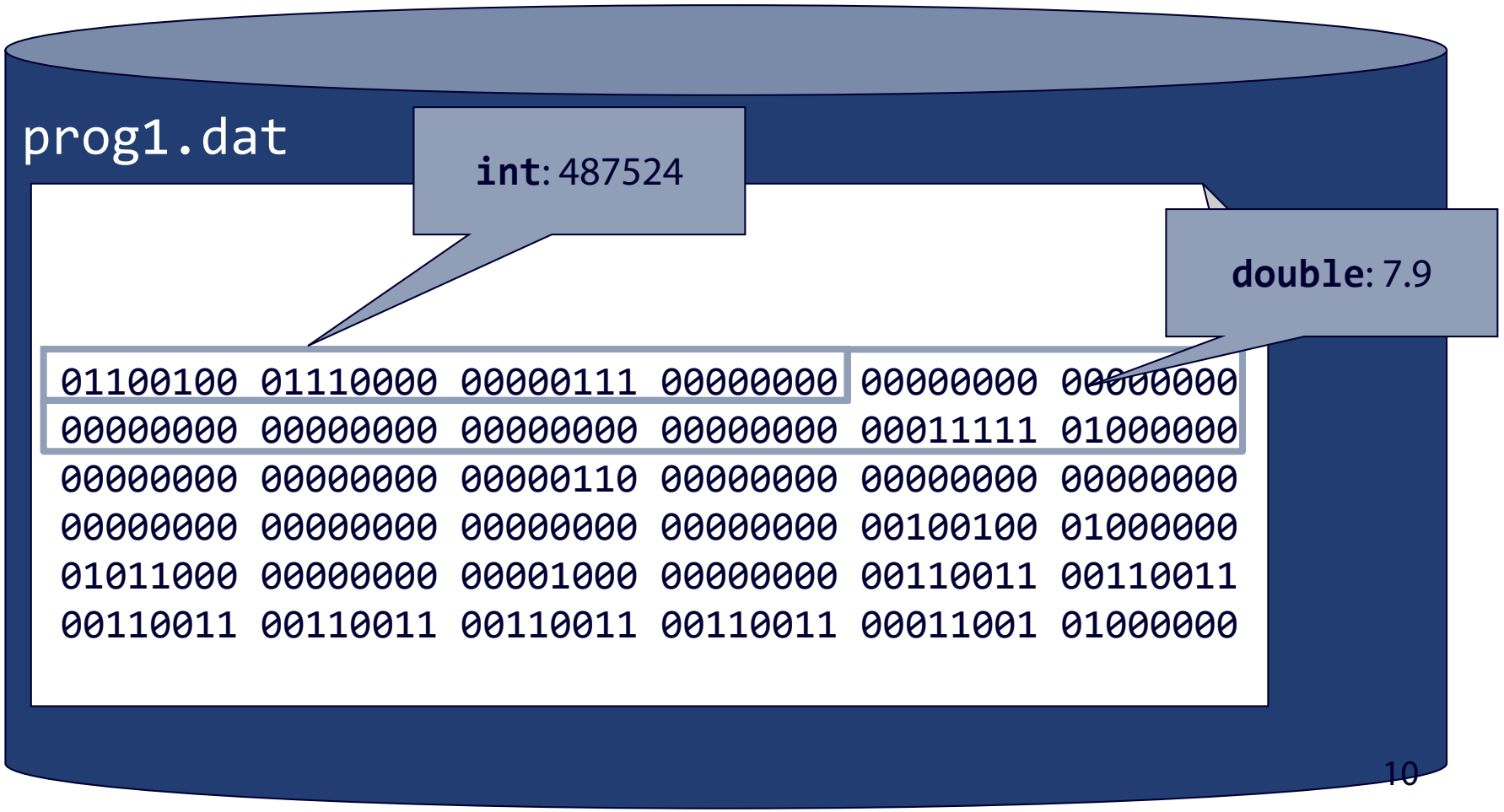
- Asociación
  - `f.open(const char sec[], ios::binary)`
- Lectura
  - `f.read(char sec[], streamsize n)`
    - `f.read(reinterpret_cast<char*>(&dato), sizeof(dato))`
- Escritura
  - `f.write(const char sec[], streamsize n)`
    - `f.write(reinterpret_cast<const char*>(&dato), sizeof(dato))`

Creación de un fichero binario

```
Introduzca un NIP (0 para acabar): 487524
Introduzca una nota: 7.9
Introduzca un NIP (0 para acabar): 454844
Introduzca una nota: 10.0
Introduzca un NIP (0 para acabar): 567896
Introduzca una nota: 6.3
Introduzca un NIP (0 para acabar): 0
```



Creación de un fichero binario



Creación de un fichero binario. Sintaxis

```
<fichero_de_notas> ::= { <nota> }
<nota> ::= <nip> <calificación>
<nip> ::= int
<calificación> ::= double
```

Creación de un fichero binario

```
/*
 * Pre:  ---
 * Post: Ha creado un fichero binario de
 *       nombre «nombreFichero» compuesto
 *       por una secuencia de pares (NIP,
 *       nota) solicitados
 *       interactivamente al operador.
 */
void crearFicheroNotas(
                const char nombreFichero[]);
```

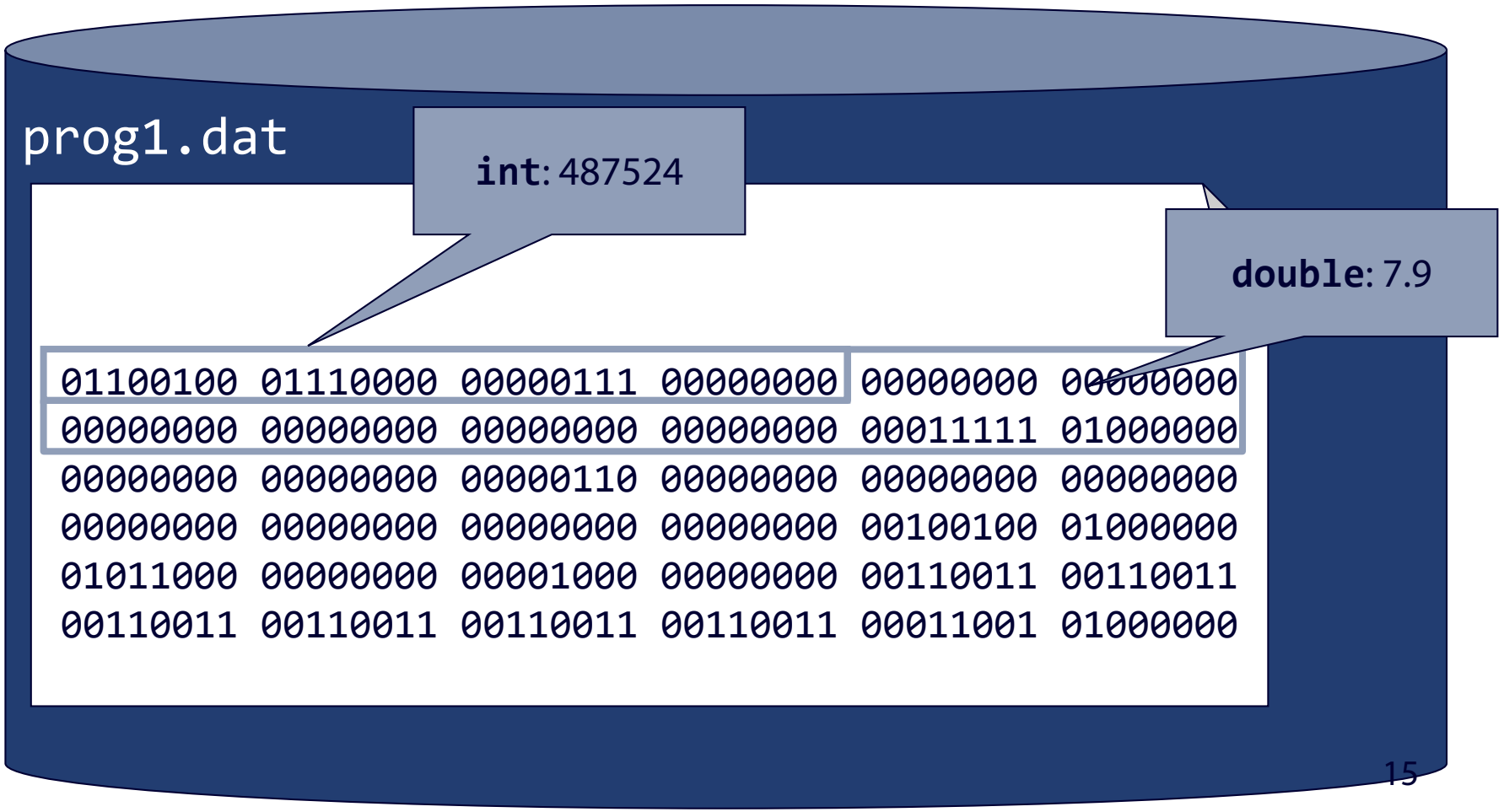
Creación de un fichero binario

```
void crearFicheroNotas(const char nombreFichero[]) {
    ofstream f(nombreFichero, ios::binary);
    if (f.is_open()) {
        cout << "Introduzca un NIP (0 para acabar): " << flush;
        int nip;
        cin >> nip;
        while (nip != 0) {
            cout << "Introduzca una nota: " << flush;
            double nota;
            cin >> nota;
            f.write(reinterpret_cast<const char*>(&nip),
                    sizeof(nip));
            f.write(reinterpret_cast<const char*>(&nota),
                    sizeof(nota));
            cout << "Introduzca un NIP (0 para acabar): "
                 << flush;
            cin >> nip;
        }
        ...
```

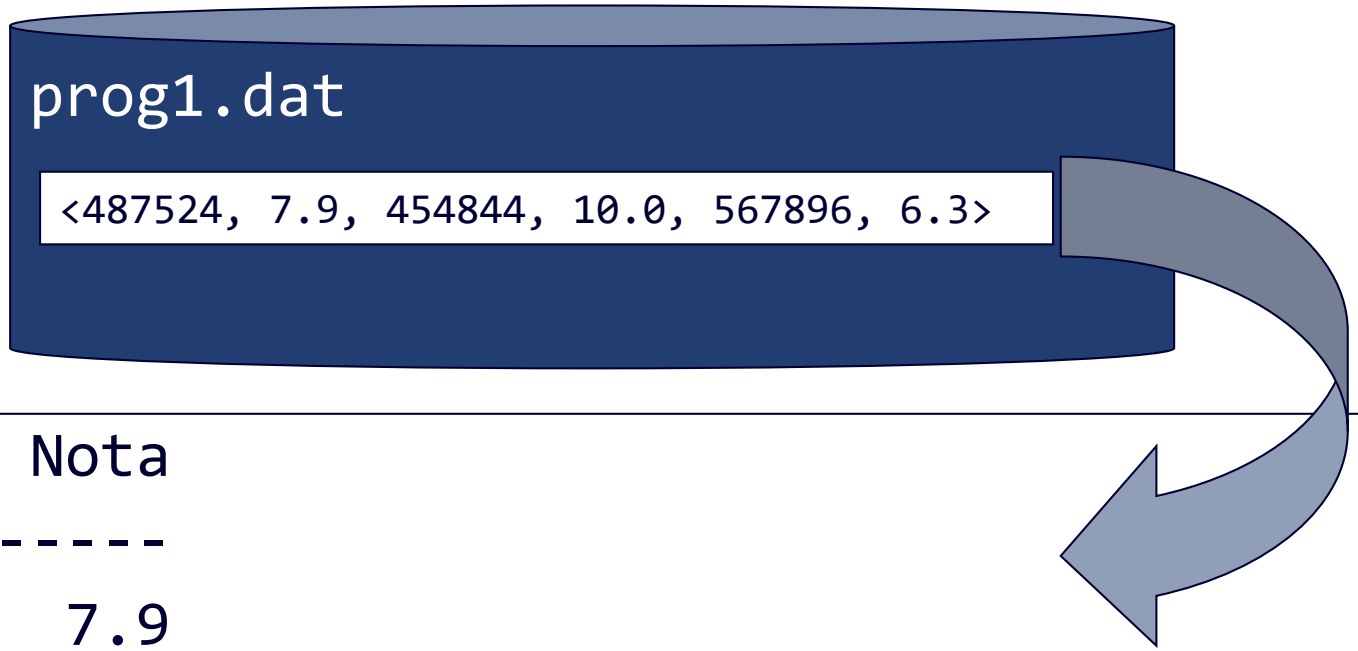
Creación de un fichero binario

```
void crearFicheroNotas(
                        const char nombreFichero[]) {
        ...
        f.close();
    }
    else {
        cerr << "No se ha podido escribir en el
"
             << "fichero \"" << nombreFichero
             << "\"" << endl;
    }
}
```

Creación de un fichero binario



Lectura de un fichero binario




```
  NIP    Nota
------------
487524    7.9
454844   10.0
567896    6.3
```

Lectura de un fichero binario. Sintaxis

```
<fichero_de_notas> ::= { <nota> }
<nota> ::= <nip> <calificación>
<nip> ::= int
<calificación> ::= double
```

Lectura de un fichero binario

```
/*
 * Pre:  «nombreFichero» es el nombre de un fichero existente
 *       binario cuya estructura consiste en una secuencia de
 *       pares (NIP, nota), de tipos int y double,
 *       respectivamente.
 * Post: Ha mostrado en la pantalla del contenido del fichero de
 *       nombre «nombreFichero», de acuerdo con el siguiente
 *       formato de ejemplo:
 *
 *         NIP   Nota
 *       ------------
 *       487524   7.9
 *       454844  10.0
 *       567896   6.3
 */
void mostrarFicheroNotas(const char nombreFichero[]);
```

Lectura de un fichero binario

```
void mostrarFicheroNotas(const char nombreFichero[]) {
    ifstream f;
    f.open(nombreFichero, ios::binary);
    if (f.is_open()) {
        cout << "  NIP   Nota" << endl;
        cout << "------------" << endl;
        cout << fixed << setprecision(1);
        int nip;
        f.read(reinterpret_cast<char*>(&nip), sizeof(nip));
        while (!f.eof()) {
            double nota;
            f.read(reinterpret_cast<char*>(&nota),
                   sizeof(nota));
            cout << setw(6) << nip << " " << setw(5) << nota
                 << endl;
            f.read(reinterpret_cast<char*>(&nip), sizeof(nip));
        }
        ...
```

Lectura de un fichero binario

```
void mostrarFicheroNotas(
                        const char nombreFichero[]) {
        ...
        f.close();
    }
    else {
        cerr << "No se ha podido leer el "
             << "fichero \"" << nombreFichero
             << "\"" << endl;
    }
}
```

Problemas con NIF

- Definición de la sintaxis de un fichero binario que almacena NIF
  - Dos versiones
- Diseño del código de dos funciones
  - Una función lee los NIF del fichero y los almacena en una tabla
  - Otra escribe en un fichero los NIF presentes en una tabla

Problemas con NIF. Sintaxis del fichero. Alternativa 2

```
<fichero-NIF-binario-alternativa-2>
                    ::= <número> { <nif> }
<número> ::= int
<nif> ::= <dni> <letra>
<dni> ::= int
<letra> ::= char
```

Problemas con NIF

```
/*
 * Pre:  n ≥ 0
 * Post: Crea un fichero de nombre «nombreFichero» en el
 *       que almacena la siguiente información codificada
 *       en binario: en primer lugar el número «n» de
 *       datos NIF que hay en «T», seguido de la secuencia
 *       de pares de datos («int», «char») que
 *       corresponden al DNI y a la letra de cada dato de
 *       tipo «Nif» almacenado en las primeras componentes
 *       de «T».
 */
void guardarNifBinAlt2(const char nombreFichero[],
                       const Nif T[], const int n);
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp

Problemas con NIF

```
void guardarNifBinAlt2(const char nombreFichero[],
                       const Nif T[], const int n) {
    ofstream f(nombreFichero, ios::binary);
    if (f.is_open()) {
        f.write(reinterpret_cast<const char*>(&n),
                sizeof(n));
        for (int i = 0; i < n; i++) {
            f.write(reinterpret_cast<const char*>(&T[i].dni),
                    sizeof(int));
            f.write(reinterpret_cast<const char*>(&T[i].letra),
                    sizeof(char));
        }
        f.close();
    }
    else {
        cerr << "No se ha podido escribir en el fichero \""
             << nombreFichero << "\"" << endl;
    }
}
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp

Problemas con NIF

```
/*
 * Pre:  El fichero de nombre «nombreFichero» almacena la
 *       siguiente información codificada en binario que
 *       representa una secuencia de NIF: En primer lugar el
 *       número de NIF almacenados en el fichero; este
 *       número es igual o mayor que cero. Le sigue una
 *       secuencia de datos de tipo «int» y «char» que
 *       representan el número de DNI y la letra de un NIF.
 * Post: Ha asignado a «nDatos» el número de NIF almacenados
 *       en el fichero y almacena en las primeras «nDatos»
 *       del vector «T» los NIF almacenados en el fichero.
 */
void leerNifBinAlt2(const char nombreFichero[], Nif T[],
                    int& nDatos);
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp

Problemas con NIF

```
void leerNifBinAlt2(const char nombreFichero[],
                    Nif T[], int& nDatos) {
    ifstream f(nombreFichero, ios::binary);
    if (f.is_open()) {
        f.read(reinterpret_cast<char*>(&nDatos),
               sizeof(nDatos));
        for (int i = 0; i < nDatos; i++)  {
            f.read(reinterpret_cast<char*>(&T[i].dni),
                   sizeof(int));
            f.read(reinterpret_cast<char*>(&T[i].letra),
                   sizeof(char));
        }
        f.close();
    }
    else {
        cerr << "No se ha podido leer del fichero \""
             << nombreFichero << "\"" << endl;
    }
}
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp

Problemas con NIF. Sintaxis del fichero. Alternativa 1

```
<fichero-NIF-binario-alternativa-1>
                    ::= <número> { <nif> }
<número> ::= int
<nif> ::= Nif
```

Problemas con NIF

```
/*
 * Pre:  n ≥ 0
 * Post: Crea un fichero de nombre «nombreFichero» en el
 *       que almacena la siguiente información codificada
 *       en binario: en primer lugar el número «n» de
 *       datos NIF que hay en «T», seguido de la secuencia
 *       de datos de tipo Nif almacenados en las primeras
 *       componentes de «T».
 */
void guardarNifBinAlt1(const char nombreFichero[],
                       const Nif T[], const int n);
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp

Problemas con NIF

```
void guardarNifBinAlt1(const char nombreFichero[],
                       const Nif T[], const int n) {
    ofstream f(nombreFichero, ios::binary);
    if (f.is_open()) {
        f.write(reinterpret_cast<const char*>(&n),
                sizeof(n));
        for (int i = 0; i < n; i++) {
            f.write(reinterpret_cast<const char*>(&T[i]),
                    sizeof(T[i]));
        }
        f.close();
    }
    else {
        cerr << "No se ha podido escribir en el fichero \""
             << nombreFichero << "\"" << endl;
    }
}
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp

Problemas con NIF

```
/*
 * Pre:  El fichero de nombre [nombreFichero] almacena la siguiente
 *       información codificada en binario que representa una secuencia
 *       de NIF: En primer lugar el número de NIF almacenados en el
 *       fichero; este número es igual o mayor que cero. Le sigue una
 *       secuencia de datos de tipo Nif.
 * Post: Asigna a nDatos el número de NIF almacenados en el fichero y
 *       almacena en T[0..nDatos-1] los NIF almacenados en el fichero
 */
void leerNifBinAlt1(const char nombreFichero[], Nif T[], int& nDatos);
```

Problemas con NIF

```
void leerNifBinAlt1(const char nombreFichero[], Nif T[], int& nDatos) {
    ifstream f;
    f.open(nombreFichero, ios::binary);
    if (f.is_open()) {
        f.read(reinterpret_cast<char*>(&nDatos), sizeof(nDatos));
        for (int i = 0; i < nDatos; i++) {
            f.read(reinterpret_cast<char*>(&T[i]), sizeof(T[i]));
        }
        f.close();
    }
    else {
        cerr << "No se ha podido leer del fichero \"" << nombreFichero
             << "\"" << endl;
    }
}
```

Código documentado en
http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/Capitulo15/fichero-nif.cpp