Codificación en lenguaje C++

- Para implementar TAD utilizaremos registros (*struct*).
  (Otra opción posible que, de momento, no usaremos: clases)

- La representación de los datos será privada (*private*) y para poder acceder a ella las operaciones del TAD serán funciones amigas (*friend*).

- Las operaciones que en pseudocódigo aparecen como procedimientos se pueden codificar con funciones *void*.
- Las operaciones que en pseudocódigo aparecen como funciones se pueden codificar con funciones que devuelven un dato.

- Los parámetros de entrada se pueden codificar con
  - Parámetros de entrada (por valor) (útiles si ocupan poca memoria)
  - Parámetros constantes por referencia (útiles si ocupan mucha memoria)
- Los parámetros de salida o de entrada/salida se codifican con
  - Parámetros por referencia

Codificación en lenguaje C++

Ejemplo de fechas: (1) archivo de cabecera (.h) (continúa…)

```
#ifndef _FECHA_H
#define _FECHA_H
// Interfaz del TAD fecha. Pre-declaraciones:

/* Los valores del TAD fecha representan fechas válidas según las reglas del calendario gregoriano
 * (adoptado en 1583) */
struct Fecha;
/* Dados los tres valores enteros dia, mes y anyo, se devuelve en f la fecha compuesta por ellos.
 * Parcial: se precisa que 1≤dia≤31, 1≤mes≤12, 1583≤anyo, y además que dia, mes y anyo formen una
 * fecha válida según el calendario gregoriano; de lo contrario, error devuelve el valor verdad*/
void crear(int dia, int mes, int anyo, Fecha& f, bool& error);
/* Devuelve el dia de la fecha */
int dia(const Fecha& f);
/* Devuelve el mes de la fecha */
int mes(const Fecha& f);
/* Devuelve el año de la fecha */
int anyo(const Fecha& f);
/* Devuelve verdad si y sólo si f1 y f2 son la misma fecha */
bool iguales(const Fecha& f1, const Fecha& f2);
/* Devuelve verdad si y sólo si la fecha f1 es cronológicamente anterior a la fecha f2 */
bool anterior(const Fecha& f1, const Fecha& f2);
/* Devuelve verdad si y sólo si la fecha f1 es cronológicamente posterior a la fecha f2 */
bool posterior(const Fecha& f1, const Fecha& f2);
...
```

↑ *tratamiento de caso de error*

Codificación en lenguaje C++

Ejemplo de fechas: (1) archivo de cabecera (.h) (… fin)

```
// Declaración:

struct Fecha {

    friend void crear(int dia, int mes, int anyo, Fecha& f, bool& error) ;
    friend int dia(const Fecha& f);
    friend int mes(const Fecha& f);
    friend int anyo(const Fecha& f);
    friend bool iguales(const Fecha& f1, const Fecha& f2) ;
    friend bool anterior(const Fecha& f1, const Fecha& f2);
    friend bool posterior(const Fecha& f1, const Fecha& f2);

    private:
      // Representación de los valores del TAD
      int elDia;
      int elMes;
      int elAnyo;
};

#endif
```

← *private ➔ encapsulación*

Codificación en lenguaje C++

Ejemplo de fechas: (2) archivo fuente (.cpp) (continúa…)

```
#include “fecha.h”

// Implementacion de las operaciones del TAD fecha

    // Operaciones auxiliares sobre enteros.

    // Devuelve verdad si y sólo si el año a es bisiesto.
    bool esBisiesto(int a) {
       return (a % 4 == 0  &&  !(a % 100 == 0 && a % 400 == 0));
    }

    // Devuelve verdad si y sólo si (d,m,a) representan una fecha válida.
    bool esFechaValida(int d, int m, int a) {
      …
    }

    // Operaciones del TAD

    void crear(int dia, int mes, int anyo, Fecha& f, bool& error) {
       …
    };

    …
```

→ ***Ver completo en el material de clase***

Codificación en lenguaje C++

Ejemplo de fechas: (2) archivo fuente (.cpp) (… fin)

```
int dia(const Fecha& f) {
    return f.elDia;
};

int mes(const Fecha& f) {
    return f.elMes;
};

int anyo(const Fecha& f) {
    return f.elAnyo;
};

bool iguales(const Fecha& f1, const Fecha& f2) {
    return f1.elDia == f2.elDia && f1.elMes == f2.elMes && f1.elAnyo == f2.elAnyo;
};

bool anterior(const Fecha& f1, const Fecha& f2) {
    …
};

bool posterior(const Fecha& f1, const Fecha& f2) {
    …
};
```

→ Ver completo en el material de clase

Codificación en lenguaje C++

Ejemplo de Tablas: (1) archivo de cabecera (.h) (continúa…)

```
#ifndef _TABLA_FREC_H
#define _TABLA_FREC_H

// Interfaz del TAD tabla de frecuencias. Pre-declaraciones:

const   int  MAX_NUM_DATOS = 1000;

struct Tabla;
void inicializar(Tabla& t);
bool anyadir(Tabla& t, int n);
int total(const Tabla& t);
int infoEnt(const Tabla& t, int n);
int infoFrec(const Tabla& t, int n);

…
```

Falta incluir el tratamiento de los casos de error.

Codificación en lenguaje C++

Ejemplo de Tablas: (1) archivo de cabecera (.h) (… fin)

```
…

// Declaración

struct Tabla {
   friend void inicializar(Tabla& t);
   friend bool anyadir(Tabla& t, int n);
   friend int total(const Tabla& t);
   friend int infoEnt(const Tabla& t, int n);
   friend int infoFrec(const Tabla& t, int n);

   // Representación de los valores del TAD
   private:

      struct Frecuencia {
               int numero;
               int frec;
      };

       Frecuencia elementos [MAX_NUM_DATOS];
       int numElementos;
};

#endif
```

Codificación en lenguaje C++

Ejemplo de Tablas: (2) archivo fuente (.cpp)

```
#include "tabla_frec.h“

// Implementacion de las operaciones del TAD

void inicializar(Tabla& t){
   t.numElementos = 0;
};

bool anyadir(Tabla& t, int n){
   …
};

int total(const Tabla& t) {
   return t.numElementos;
};

int infoEnt(const Tabla& t, int n) {
   …
};

int infoFrec(const Tabla& t, int n) {
   …
};
```

→ Ver completo en el material de clase