FASE I. COMPRENSIÓN DEL MATERIAL PROPORCIONADO

- Descargar los ficheros fuente.
- Compilarlos y ejecutarlos:

```
$ g++ -g main_sparse_vector_t.cpp -o
main_sparse_vector_t
$ ./main_sparse_vector_t < data_sparse_vector_t.txt
```

- Examinar los ficheros de cabecera identificando los constructores, el destructor, los métodos para acceder a los atributos, los métodos para lectura desde teclado y escritura a pantalla, e identificar los elementos privados y públicos, tanto atributos como métodos. Identificar la sintaxis de las plantillas. Establecer dónde se efectúa la sobrecarga de operadores, determinar qué operadores han sido sobrecargados.
- Comprender la clase **`pair_t`** que implementa una pareja índice y valor para cualquier tipo (numérico) T.
- Comprender la definición de la clase vector disperso (**`sparse_vector_t`**).
- Comprender la definición de la nueva clase **`pair_vector_t`** como:

```
typedef vector_t<pair_t<double>> pair_vector_t;
```

FASE II. DESARROLLO DE UN CONSTRUCTOR PARA LA CLASE `sparse_vector_t`

Desarrollar el **constructor** de la clase **`sparse_vector_t`** tal y como se ha especificado durante la clase de problemas para una clase vector disperso implementado sobre un vector de pares de tipo **`double`** (**`vector_t<pair_t<double>>`**).

FASE III. DESARROLLO DE UN MÉTODO PARA CÁLCULO DEL PRODUCTO ESCALAR ENTRE VECTOR DISPERSO Y VECTOR DENSO

Desarrollar un método que efectúe el **producto escalar** entre el vector disperso invocante y un vector denso que se pasa como parámetro. El método debe tener la siguiente cabecera:

```
double scal_prod(const vector_t<double>&);
```

FASE IV. DESARROLLO DE UN MÉTODO PARA EL CÁLCULO DEL PRODUCTO ESCALAR ENTRE VECTORES DISPERSOS

Desarrollar un método que efectúe el **producto escalar** entre el vector disperso invocante y otro vector disperso que se pasa como parámetro. El método debe tener la siguiente cabecera: `double scal_prod(const sparse_vector_t&);`

Resultado esperado

Si ejecutan la práctica con todas las fases desarrolladas, la salida esperado por pantalla sería:

```
v1= 10: [ 0 0 0 3.4 0 5.6 0 0 8.9 0 ]
v2= 10: [ 0 1.2 0 0 4.5 0 0 0 0 9.1 ]
v3= 10: [ 0 1.3 0 3.5 4.6 5.7 0 0 8.1 9.2 ]

sv1= 10(3): [ (3:3.4) (5:5.6) (8:8.9) ]
sv2= 10(3): [ (1:1.2) (4:4.5) (9:9.1) ]
sv3= 10(6): [ (1:1.3) (3:3.5) (4:4.6) (5:5.7) (8:8.1) (9:9.2) ]

sv3 * v1= 115.91
sv3 * sv2= 105.98
```