Práctica II

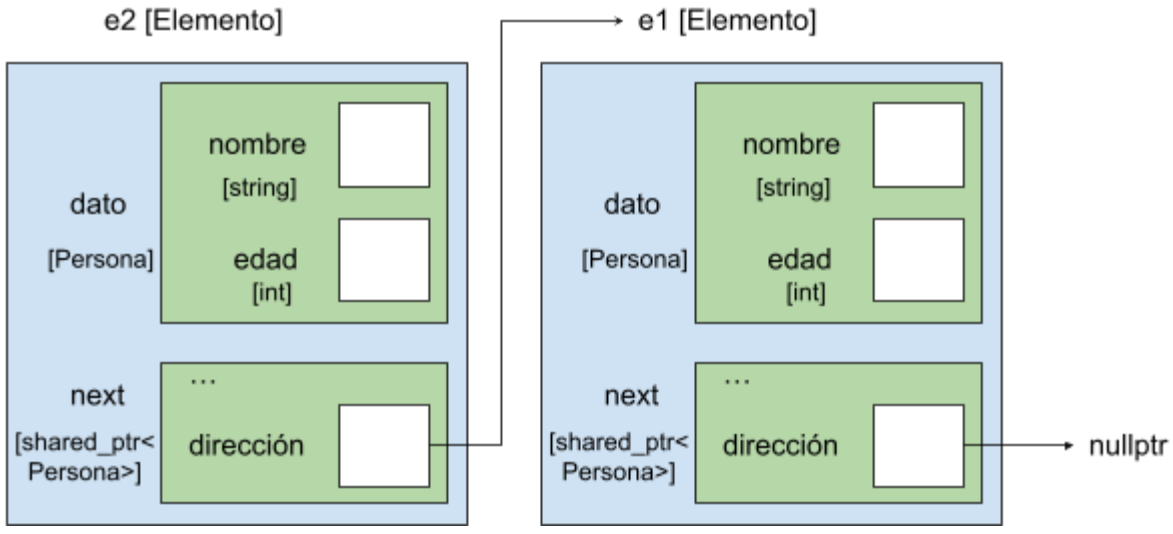
Todos debéis entregar un único fichero [practica2.zip] antes de la finalización de la práctica. Al comienzo del mismo debéis incluir un comentario con vuestro nombre y el de vuestro compañero en caso de realizar la práctica en pareja.

Recordad que las prácticas se realizan en vuestra sala de grupo/individual (según la modalidad escogida) siguiendo el procedimiento habitual (compartir audio, webcam, escritorio y grabando la sesión). Y que podéis realizar consultas en internet, apuntes, campus, etc. No estando permitido herramientas de comunicación fuera del grupo de trabajo.

Cualquier incumplimiento supondrá la descalificación directa en la práctica.

Vamos a implementar un tipo de datos para representar una fila de personas que están esperando turno en una panadería.

En la siguiente imagen podéis ver una representación gráfica de una fila de dos elementos como ejemplo. El primer elemento, el que espera a ser atendido en primer lugar, sería e1. Como no tiene nadie delante, el puntero a Persona que encapsula apunta a nullptr.
El puntero a persona del siguiente elemento en la lista, el que ocupa la segunda posición, apuntará a la dirección de e1. Y así sucesivamente para el resto de integrantes que pudiera tener la fila de la panadería (ten en cuenta que no hay límite máximo de personas y que en la fila puede que no haya personas).



A partir de las declaraciones de las clases que disponéis a continuación debéis implementar los métodos indicados en la clase Fila. Cada uno tiene una puntuación asociada

```
struct Persona{
	std::string nombre;
	int edad{0};
};

struct Elemento{
	Elemento(const Persona &p){dato=p;}
	Persona dato;
	shared_ptr<Elemento> next{nullptr};
};

class Fila{
	public:
		int size() const; //Devuelve el tamaño de la fila [2 puntos]
		bool empty() const; //Devuelve true si la fila está vacía [1 punto]
		Persona getFront() const; //Devuelve primera persona de la fila [2 puntos]
		Persona getBack() const; //Devuelve última persona de la fila [0,5 puntos]
		bool PersonPresent(const Persona& dato); //Busca una persona en la fila,
	devolverá true si está y false si no está [2 puntos]
		//Para el método PersonPresent necesitarás sobrecargar el operador == para
	comparar las personas [1,5 puntos]
		void push(const Persona& dato); //Coloca persona en la fila [1 punto]
	private:
		std::shared_ptr<Elemento> _ultimo = nullptr;
};
```

Rúbrica de evaluación

El programa no compila o no se asemeja a lo que se pide	0%
El programa no hace lo que se pide pero el código es correcto y se aproxima a lo pedido	40%
El programa funciona correctamente	100%