Memoria

El objetivo de este control es implementar un TAD genérico sencillo, así como familiarizarse con la gestión de memoria dinámica en clases C++

1) Trabajo a realizar

Debe implementarse un TAD genérico *Memoria*<T> que simule una memoria ilimitada en la que, en cada una de sus celdas, puede almacenarse un valor de tipo T. Las celdas comienzan a direccionarse desde 0, y puede utilizarse cualquier dirección no negativa. Dicho TAD proporcionará las siguientes operaciones:

- Un constructor `Memoria(v)` que construya una memoria en la que, por defecto, cada celda almacene el valor *v*.
- Una operación observadora `fetch` que tome como argumento una dirección *d*, y devuelva el valor almacenado en dicha dirección. La dirección debe ser mayor o igual que 0 (en caso contrario, se levanta la excepción `DireccionNoValida`).
- Una operación mutadora `load` que tome como argumento una dirección *d* y un valor *v*, y almacene *v* en la dirección *d*. La dirección debe ser mayor o igual que 0 (en caso contrario, se levanta la excepción `DireccionNoValida`).

Como representación debe gestionarse directamente un array dinámico (no se permitirán soluciones que utilicen las clases y plantillas de la STL -tales como `vector`-, implementaciones de otros TADS, etc.). Inicialmente dicho array tendrá un tamaño de `2` elementos, que contendrán el valor por defecto. Si se consulta una dirección válida (es decir, no negativa) *d* fuera del rango del array con `fetch`, deberá devolverse el valor por defecto. Si se almacena un valor *v*, distinto del valor por defecto, en una dirección *d* fuera del rango del array con `load`, el array deberá redimensionarse para que tenga exactamente *d*+1 elementos. El elemento *d* deberá contener *v*, y el resto de nuevos elementos añadidos deberán contener el valor por defecto. Dado que la implementación estará basada en el manejo de memoria dinámica, deberán incluirse los constructores y métodos adicionales necesarios para garantizar el correcto funcionamiento, así como para evitar cualquier pérdida de memoria.

2) Código de apoyo

Se proporcionan los siguientes archivos:

- `Memoria.h`. Este archivo debe completarse con la implementación del TAD genérico pedido.
- `main.cpp`. Programa de prueba. Este archivo no debe modificarse. El programa mantiene una memoria en la que los valores almacenados son enteros. El programa comienza leyendo el valor por defecto almacenado en las celdas de la memoria. A continuación, lee y ejecuta *comandos*. Los comandos leídos son de los siguientes tipos:
  - `load` *d v*: Almacena el valor *v* en la dirección *d*. Imprime OK en caso de que la dirección sea válida, DIRECCION_INVALIDA en otro caso.
  - `fetch` *d*: Recupera el valor almacenado en la posición *d*. Imprime el valor recuperado en caso de que la dirección sea válida, DIRECCION_INVALIDA en otro caso.
  - `termina`. Termina la ejecución.

  A continuación, se muestra un ejemplo de entrada / salida:

Entrada	**Salida**

`-1`
`load 5 3`
`fetch 4`
`fetch -1`
`load 6 -100`
`fetch 6`
`fetch 5`
`termina`
	`OK`
	`-1`
	`DIRECCION_INVALIDA`
	`OK`
	`-100`
	`3`