ARRAY

El ejercicio propuesto consiste en desarrollar una plantilla de clase llamada Array en C++98, que gestione elementos de tipo T y que implemente una serie de comportamientos y funciones específicas. A continuación, se detallan los aspectos clave del ejercicio:

Descripción del Ejercicio:

1. **Constructores:**
    a. **Constructor sin parámetros:** Crea un arreglo vacío.
    b. **Constructor con un parámetro de tipo unsigned int n:** Crea un arreglo de n elementos inicializados por defecto. Por ejemplo, al compilar int *a = new int(); y mostrar *a, se observa que a apunta a un entero inicializado a cero.
2. **Constructor de copia y operador de asignación:**
    a. Ambos deben permitir que, tras la copia, modificar el arreglo original o su copia no afecte al otro.
3. **Gestión de memoria:**
    a. Se debe utilizar operator new[] para la asignación de memoria.
    b. No se permite la asignación preventiva de memoria; es decir, no se debe asignar memoria por adelantado.
    c. El programa nunca debe acceder a memoria no asignada.
4. **Acceso a elementos:**
    a. Los elementos deben ser accesibles mediante el operador de subíndice [ ].
    b. Si se accede a un índice fuera de los límites, se debe lanzar una excepción de tipo std::exception.
5. **Función miembro size():**
    a. Devuelve el número de elementos en el arreglo.
    b. No recibe parámetros y no debe modificar la instancia actual.
6. **Archivo main.cpp:**
    a. Debe contener pruebas que aseguren que todo funciona como se espera.

Propósito de Aprendizaje:

Este ejercicio tiene como objetivo profundizar en varios conceptos fundamentales de C++98:

- **Plantillas de clases:** Aprender a crear clases genéricas que puedan manejar diferentes tipos de datos.
- **Gestión dinámica de memoria:** Utilizar new[] y delete[] para asignar y liberar memoria de forma segura.
- **Manejo de excepciones:** Implementar mecanismos para manejar errores, como el acceso fuera de los límites del arreglo.
- **Operadores sobrecargados:** Sobrecargar operadores, como el de subíndice [ ], para proporcionar una sintaxis intuitiva al usuario de la clase.
- **Regla de tres:** Implementar correctamente el constructor de copia, el operador de asignación y el destructor para asegurar una gestión adecuada de recursos.

Enfoque para Resolver el Ejercicio en C++98:

1. **Definición de la Plantilla de Clase:**
   a. Declarar la plantilla de clase Array que gestione elementos de tipo T.
   b. Incluir un puntero a T para almacenar los elementos y una variable para el tamaño del arreglo.
2. **Constructores:**
   a. **Constructor por defecto:** Inicializar el puntero a NULL y el tamaño a 0.
   b. **Constructor con parámetro unsigned int n:** Asignar memoria para n elementos utilizando new[] y asegurarse de que cada elemento esté inicializado por defecto.
3. **Constructor de Copia y Operador de Asignación:**
   a. **Constructor de copia:** Crear un nuevo arreglo copiando los elementos del arreglo original.
   b. **Operador de asignación:** Liberar la memoria existente, asignar nueva memoria y copiar los elementos del arreglo fuente.
   c. En ambos casos, garantizar que las modificaciones en uno de los arreglos no afecten al otro.
4. **Destructor:**
   a. Liberar la memoria asignada utilizando delete[] para evitar fugas de memoria.
5. **Operador de Subíndice [ ]:**
   a. Sobrecargar el operador para permitir el acceso a los elementos del arreglo.
   b. Comprobar si el índice está dentro de los límites; si no, lanzar una excepción de tipo std::exception.
6. **Función size():**
   a. Devolver el número de elementos en el arreglo.
   b. Declarar esta función como const para asegurar que no modifica la instancia actual.

Recursos Recomendados:

Para abordar con éxito este ejercicio, se recomiendan los siguientes recursos:

1. **"Curso C++ No Tan Básico. Sobrecarga de operadores con plantillas"**
   Descripción: Este video del curso "C++ No Tan Básico" aborda cómo realizar sobrecargas de operadores en objetos definidos por el usuario utilizando plantillas. Es especialmente útil para entender la implementación del operador de subíndice [ ] en una clase plantilla.

   Enlace: https://www.youtube.com/watch?v=o5Y21HBIlrs

2. **"013.- Curso de C++ Moderno. Sobrecargar el Operador New"**
   Descripción: En este videotutorial del "Curso de C++ Moderno", se explica cómo sobrecargar el operador new. Aunque el ejercicio no requiere la sobrecarga de este operador, el video proporciona una comprensión profunda sobre la asignación dinámica de memoria en C++, lo cual es esencial para implementar correctamente la gestión de memoria en la clase Array.

   Enlace: https://www.youtube.com/watch?v=exGdwtLD5Ig

3. **"C++ Templates y Excepciones"**
   Descripción: Este video ofrece una introducción a las plantillas y al manejo de excepciones en C++. Aunque se centra en C++ moderno, los conceptos fundamentales son aplicables a C++98 y te ayudarán a implementar la plantilla de clase Array y el manejo de excepciones al acceder a elementos fuera de los límites.

   Enlace: https://www.youtube.com/watch?v=GwZLa90V5Qo

4. **"Curso de C++ Moderno. Expresiones y sobrecarga de Operadores"**
   Descripción: Este videotutorial profundiza en el uso de expresiones y la sobrecarga de operadores en C++. Aunque se enfoca en C++ moderno, los principios de sobrecarga de operadores son aplicables a C++98 y serán útiles para implementar el operador de subíndice [ ] en la clase Array.

   Enlace: https://www.youtube.com/watch?v=NlH4MM5w5xI

5. **"144.- Curso C++ No Tan Básico. Sobrecarga de operadores unarios"**
   Descripción: En este video del curso "C++ No Tan Básico", se aborda la sobrecarga de operadores unarios como +, - y !. Aunque el ejercicio se centra en la sobrecarga del operador de subíndice, este recurso te ayudará a comprender mejor cómo funciona la sobrecarga de operadores en general en C++.

   Enlace: https://m.youtube.com/live/zuAu6wP9HCk

Estos recursos te proporcionarán una comprensión sólida de los conceptos clave necesarios para abordar el ejercicio, incluyendo plantillas de clases, gestión dinámica de memoria, manejo de excepciones y sobrecarga de operadores en C++98.

El código

Explicación del Código

El código define y prueba una **clase plantilla** Array<T> en **C++98**, que representa un **arreglo dinámico** de elementos de tipo T. Este arreglo tiene características similares a std::vector, pero con una implementación manual del **manejo de memoria dinámica**. Se implementan constructores, un operador de asignación, sobrecarga del operador de subíndice [ ], y una clase interna de excepción para manejar accesos fuera de los límites.

A continuación, se desglosan los puntos principales del código:

1. Definición de la Clase Array<T> en Array.hpp

1.1 Atributos Principales de la Clase

- T* store: Puntero a un bloque de memoria dinámico donde se almacenan los elementos del arreglo.
- uint32_t storeLen: Número de elementos en el arreglo.

1.2 Constructores

- **Constructor por defecto (Array())**
  - Crea un arreglo vacío (store apunta a una memoria de tamaño 0).
  - Imprime "Array - default constructor called".
- **Constructor con tamaño (Array(uint32_t n))**
  - Crea un arreglo de n elementos inicializados por defecto.
  - Imprime "Array - parametric constructor called".
- **Constructor de copia (Array(const Array &src))**
  - Crea una copia del arreglo original.
  - Copia los valores del otro arreglo.
  - Imprime "Array - copy constructor called".

1.3 Operador de Asignación (operator=)

- **Evita fugas de memoria**:
  - Libera (delete[]) la memoria previamente asignada.
  - Reserva una nueva memoria (new[]).
  - Copia los elementos uno por uno.
- Imprime "Array - assignation operator called".

1.4 Operador de Subíndice [ ] (operator[])

- Permite acceder a elementos mediante arr[i].
- Si el índice está fuera del rango, **lanza una excepción** std::exception.

1.5 Método size()

- Retorna la cantidad de elementos del arreglo (storeLen).

1.6 Clase Interna de Excepción OutOfLimits

- Se lanza cuando se intenta acceder fuera de los límites del arreglo.
- Devuelve el mensaje "Error: Out of limits".

1.7 Destructor (~Array())

- Libera la memoria reservada con delete[].
- Imprime "Array - destructor called".

2. Código main.cpp - Pruebas de la Clase Array<T>

2.1 Creación de Arrays

- Se crean dos arreglos de distintos tamaños:
  Array<int> a(3); Array<int> b(5);
    - a almacena 3 elementos.
    - b almacena 5 elementos.

2.2 Mostrando Valores por Defecto

- Se imprimen los valores iniciales de a y b, que deberían ser 0.

2.3 Modificación de Elementos

- Se asignan valores a a: a[0] = 1;
  a[1] = 1;
  a[2] = 1;

- b no se modifica, por lo que seguirá con valores predeterminados (0).

2.4 Uso del Operador de Asignación (b = a)

- b = a; copia los valores de a en b, pero creando una copia independiente.

2.5 Acceso Fuera de los Límites

- Se intenta acceder a un índice inválido de a:

```
try {
  cout << a[a.size() + 1] << endl;
} catch (std::exception &e) {
  cerr << RED << e.what() << RESET << endl;
}
```

- Se lanza una excepción "Error: Out of limits".

3. Puntos Clave y Conceptos Aplicados

1. **Uso de plantillas (template <typename T>)**
   a. Permite crear arreglos de cualquier tipo (int, double, std::string, etc.).
2. **Manejo de memoria dinámica (new[] y delete[])**
   a. new[] asigna memoria en el **heap**.
   b. delete[] evita **fugas de memoria**.
3. **Regla de Tres en C++98**
   a. Implementación de:
      i. **Constructor de copia** (Array(const Array &src))
      ii. **Operador de asignación** (operator=)
      iii. **Destructor** (~Array())
4. **Sobrecarga de operadores (operator[] y operator=)**
   a. operator[] permite acceder a elementos.
   b. operator= copia arreglos correctamente.
5. **Manejo de excepciones (throw std::exception)**
   a. Lanza una excepción cuando se accede a un índice fuera del límite.

Resumen

- El código define una clase plantilla Array<T> para manejar arreglos dinámicos en C++98.
- Se implementan los constructores, el operador de asignación y la sobrecarga del operador [ ].
- Se utiliza memoria dinámica con new[] y delete[].
- Se maneja el acceso fuera de los límites con excepciones (std::exception).
- En main.cpp, se crean y manipulan arreglos de distintos tamaños para probar la clase.