RETO 2: ABSTRACCIÓN

ÍNDICE

Realizado por: Arturo Alonso Carbonero y José Luis Molina Aguilar
Grupo: 2ºC

1. Introducción

El reto consiste en construir e indicar la especificación de los tipos de dato abstractos necesarios para la creación del clásico juego del Tetris.

Para realizar dicho proyecto, hemos creado un tipo de dato “matrizGeneral”, de tipo int(entero), que nos permitirá construir tanto el tablero o acumulador, donde las piezas van colocándose, como las propias piezas.
Por otra parte, hemos creado un tipo “Cola” basado en la implementación de las colas de la *stl* de tipo genérico, esto es, el tipo de los objetos de la cola será el que se le indique en la función principal.

A continuación, se mostrará una breve explicación de cada tipo de dato, así como su especificación, el funcionamiento de algunos de los métodos de los que disponen y un ejemplo del procedimiento de la función main.

2. Tipos abstractos.

2.1. Tipo matrizGeneral.

Se trata de un tipo que crea objetos, concretamente matrices, del tamaño que se le indique. Permite la creación del acumulador de piezas y de las propias piezas, así como su manipulación y su interacción con el contenedor de piezas de tipo Cola. Esta clase contiene los métodos necesarios para que el jugador pueda interactuar con la pieza que actualmente se esté jugando en el tablero.

2.2.1. Especificación del tipo matrizGeneral.

```
#ifndef MATRIZ_H
#define MATRIZ_H

#include <iostream>

using namespace std;

class matrizGeneral{
private:
  int nfilas, ncols;
  int **matrizGeneral;
  int tipoPieza, nFilasEliminadas;
  bool puedeDesplazar, piezaColocada;
public:
  /**
    * @brief Constructor por defecto de la clase, que creará una matriz vacía
    */
  matrizGeneral();

  /**
    * @brief Constructor de la clase
    * @param nfilas número de filas de la matriz
    * @param ncols nmero de columnas de la matriz
    * @param tipoPieza número que indica si se creará el tablero o una de las siete posibles piezas
    * @return Crea un tablero/pieza de nfilas * ncols
    * @pre nfilas debe ser un número >= 0
    * @pre ncols debe ser un número >= 0
    * @pre tipoPieza debe ser un número entre 0 y 7
    * @note sirve para crear tanto las piezas como el tablero, si tipo=0 => crea tablero
    */
  matrizGeneral(int nfilas, int ncols, int tipoPieza);

/**
    * @brief Destructor
    */
  ~matrizGeneral();

/**
    * @brief Getter de nfilas
    * @return Devuelve el numero de filas
    */
  int getNfilas();
```

```
/**
   * @brief Getter de ncols
   * @return Devuelve el numero de ncols
   */
  int getNcols();
  /**
   * @brief Getter del tipo de pieza
   * @return delvuelve el tipo de la pieza actual
   */
  int getTipoPieza();

  /**
   * @brief Modifica el valor de la variable que indica si se ha colocado la pieza
   * @param Valor booleano que se desee asignar
   * @return True si se ha colocado la pieza al final del tablero
   */
  bool setPiezaColocada(bool colocada);

/**
   * @brief Comprueba si hay filas completas cuando la pieza es colocada
   * @param tablero El tablero en su estado actual
   * @return Devuelve true si se pueden eliminar filas
   */
  bool hayLineasCompletas(matrizGeneral &tablero);

  /**
   * @brief Elimina las lineas posibles
   * @param El tablero en su estado actual
   */
  void eliminarFilas(matrizGeneral &tablero);

  /**
   * @brief Comprueba si la pieza actual se ha colocado
   * @param La pieza actual
   */
  bool piezaColocada(matrizGeneral &pieza);

/**
   * @brief Extrae la primera pieza de la cola de piezas
   * @param Contenedor de piezas en su estado actual
   * @note Coloca la pieza en el parte superior y central del tablero, llamando a la funcion insertar
de la cola
   */
  void introducirPieza(matrizGeneral &pieza);

  /**
   * @brief Rota la pieza actual
   */
  void rotarPieza()
```

```
/**
   * @brief Baja la pieza actual en el tablero una posición
   */
  void bajarPieza();

  /**
   * @brief Indica si el tablero está completo
   * @returm True si esta lleno, esto es, no se pueden insertar más piezas
   */
  bool tableroCompleto();

  /**
   * @brief Desplaza la pieza actual según se le indique
   * @param I o D, izquierda o derecha
   */
  void desplazarPieza(char izquierdaDerecha);

  /**
   * @brief Muestra el estado actual de la partida
   */
  void mostrarEstado();
};
#include "Matriz.cpp"

#endif
```

2.2. Tipo Cola general.

Se trata de una cola que admite objetos de cualquier tipo, basada en la implementación de la *queue* de la *stl*, añadiendo ciertas funcionalidades necesarias para este proyecto en concreto.
El objeto contenedor del tipo Cola solo interactuará con la clase matrizGeneral una vez se haya creado, el jugador solo podrá visualizar su estado.

2.2.1. Especificación del tipo Cola general.

```
#ifndef COLA_H
#define COLA_H

#include <iostream>

using namespace std;

template <class C>
class Cola{
private:
  C *cola;
  int tipoPieza;
  int tamCola;
public:
  /**
    * @brief Constructor de la clase
    * @param Tamaño fijo del contenedor
    */
  Cola(int tam);

  //NOTA -> Nunca se creará ni se copiará

  /**
    * @brief Destructor
    */
  ~Cola();

  /**
    * @brief Construye una pieza de tipo aleatorio
    * @param Tipo de la pieaza (Aleatorio entre 1 y 7)
    * @param Filas de la pieza
    * @param Columnas de la pieza
    * @return Devuelve una pieza
    * @note Llama al constructor de la clase matrizGeneral y le indica el tipo de pieza
    * @note Además, llama a insertar, de esta clase, e inserta la pieza creada
    */
  matrizGeneral construirPieza(int fil, int col, int tipoPieza);

/**
    * @brief Insertar una nueva pieza
    * @param Objeto a insertar
    */
  void insertar(const C pieza);
```

```
/**
   * @brief Sacar la primera pieza de la cola
   */
  void extraerPieza();

  /**
   * @brief Frente de la cola de piezas
   */
  C frente() const;

  /**
   * @brief Muestra el esatdo actual del contenedor de contenedor de piezas
   */
  void mostrarContenedor();
};
#include "Cola.cpp"

#endif
```

3. Función principal main, procedimiento.

Tras la creación de los objetos y variables pertinentes, en el main se realizará lo siguiente:

Dentro de un blucle de tipo *while*, cuya condición será si la pieza actual ha sido colocada en el acumulador y una variable booleana que indicará si se ha finalizado el juego:

1- El tablero retirará la primera pieza del contenedor y la insertará en su parte superior central. El contenedor por su parte, eliminará dicha pieza e introducirá una nueva, al final de la cola, de cualquiera de los posibles tipos de piezas. Se mostrará además el estado del juego.

2- Se entrará en otro bulce igual que el principal (únicamente tendrá en cuenta si la pieza se ha colocado, no si el juego ha finalizado), donde la pieza bajará en cada iteración hasta ser colocada. En cada una de dichas iteraciones, se dará la posibilidad al jugador de rotar o desplazar, si es posible, la pieza actual.

3- Una vez la pieza haya alcanzado el final del acumulador, se comprueba si se pueden eliminar filas del mismo y, si es así se eliminan y se actualiza el marcador.
Se preguntará al jugador si desea finalizar el juego. Si indica que desea terminar, el bucle principal finaliza y se muestra el estado final del acumulador, del contenedor y del marcador. Si, por el contrario, el jugador indica que desea continuar, el bucle principal volverá a iterar todo el procedimiento anterior.

3.1. Especificación.

```
#include <iostream>
#include "matrizGeneral.h"
#include "Cola.h"
//Para poder generar números aleatorios:
#include <stdlib.h>
#include <time.h>

using namespace std;

int main(){
  int tipoPieza=1+rand()%(8-1); // Número aleatorio entre 1 y 7
  int tamContenedor=7;
  const int filasPieza=4, colsPieza=4;
  bool finDelJuego=false;

  matrizGeneral tablero=matrizGeneral(20, 40, 0);
  // 0 indica el tipo de la pieza creada, en este caso 0 se corresponde al tablero
  Cola<matrizGeneral> contenedor(tamContenedor);
  matrizGeneral pieza;

...
  //Prodcedimiento indicado
...

  return 0;
}
```

```
/* NOTA
 Para insertar piezas en la cola:
   pieza=contenedor.construirPieza(filasPieza, colsPieza, tipoPieza);
 Para extraer la pieza de la cola y colocarla en el tablero:
   pieza=contenedor.frente();
   tablero.introducirPieza(pieza);
   contenedor.extraerPieza();
   //Se crea y se introduce una nueva pieza igual que en el anterior ejemplo
 */
```

FIN.