Documentacion FloatCell

FloatCell.h

```
#ifndef TEMA1_FLOATCELL_H
#define TEMA1_FLOATCELL_H

class FloatCell {
public:
    //constructor implicito
    explicit FloatCell(float newValue = 0); //Redactamos el valor que queremos, en
este caso es 0

    //constructor wxplicito por copia
    FloatCell(const FloatCell &rhs); //declaramos el valor a constante

    //explicit constructor por referencia
    FloatCell(FloatCell &&rhs) noexcept; //lo declaramos para que no haya
excepciones

    //Destructor, activated when free is used
    ~FloatCell() = default;

    //Asignación por copia, para Rvalores
    FloatCell &operator = (const FloatCell &rhs);
    FloatCell &operator = (FloatCell &&rhs) noexcept;

    //Nos dice con que tipo de datos vas a trabajar en este caso "float" Porque
solo puede trabajar con datos de su propia clase
    FloatCell &operator = (float rhs);

    //esta función sirve ver qué valor se asigna
    float getValue() const;

    //esta funcion es para asignar un valor
    void setValue(float newValue);

private:
    //Declaramos la variable float
    float storedValue;
};


#endif //TEMA1_FLOATCELL_H
```

FloatCell.cpp

```
#include "FloatCell.h"
```

```
//implicit constructor
FloatCell::FloatCell(float newValue) : storedValue(newValue){} //hacemos el
contructor implicito

//Copy constructor
FloatCell::FloatCell(const FloatCell &rhs) : storedValue(rhs.storedValue){}
//hacemos el constructor tipo copia

//Move constructor
FloatCell::FloatCell(FloatCell &&rhs) noexcept : storedValue(rhs.storedValue) {
    rhs.storedValue = 0;
} //hacemos el constructor tipo movimiento


FloatCell &FloatCell::operator=(const FloatCell &rhs) {
    if(this != &rhs)
        storedValue = rhs.storedValue;

    return * this;
} //verifica si StoredValue es diferente a la variable elegida, en caso de ser
diferente, los iguala

FloatCell &FloatCell::operator=(FloatCell &&rhs) noexcept {
    if (this != &rhs) {
        storedValue = rhs.storedValue;
        rhs.storedValue = 0;
    }

    return *this;
} // en esta parte es para remover o eliminar lo innecesario sin excepciones

void FloatCell::setValue(float newValue)
{
    storedValue = newValue;
}
float FloatCell::getValue() const
{
    return storedValue;
}
```