CPP – CR : Template

Robin DELEPINE – Romain POURIAS

SEC 2024 : 08-12-2022

TP : Template

OBJECTIF :

Manipuler des classes, de travailler sur la surcharge d'opérateurs et de mettre en œuvre le polymorphisme et utilisant des Template.

PARTIE 1 : Code Point.h

```cpp
#pragma once
#include <iostream>
#include <vector>
using namespace std;

//----------------------------------------
// declaration de la classe
//----------------------------------------

template<typename T>
class point
{
protected:
    T x = 0;
    T y = 0;
public :
    point();
    point(T x, T y);
    void translater(T a, T b);
    point(point const & pt);
    T getX();
    T getY();
    void setX(T a);
    void setY(T b);
};

//----------------------------------------
// declaration des fonctions de la classe
//----------------------------------------

template<typename T>
inline point<T>::point()
{
    this->x = 0;
    this->y = 0;
}

template<typename T>
inline point<T>::point(T x, T y)
{
    this->x = x;
    this->y = y;
}

```


```cpp
#pragma once
#include <iostream>
#include <vector>
using namespace std;

//----------------------------------------
// declaration de la classe
//----------------------------------------

template<typename T>
class point
{
protected:
    T x = 0;
    T y = 0;
public :
    point();
    point(T x, T y);
    void translater(T a, T b);
    point(point const & pt);
    T getX();
    T getY();
    void setX(T a);
    void setY(T b);
};

//----------------------------------------
// declaration des fonctions de la classe
//----------------------------------------

template<typename T>
inline point<T>::point()
{
    this->x = 0;
    this->y = 0;
}

template<typename T>
inline point<T>::point(T x, T y)
{
    this->x = x;
    this->y = y;
}
```


PARTIE 2 : Forme.h

```hpp
#pragma once
#include "point.h"
template<typename S>
class forme
{
protected:
    vector<point<S>> vectorPt;
public:
    forme();
    forme(point<S> pt);
    vector<point<S>> getVecteurPoint();
    void setVectorPoint(vector<point<S>> vector);
    void addPoint(point<S> pt);

    virtual S perimetre() = 0;
    virtual S surface() = 0;
};

template<typename S>
inline forme<S>::forme()
{
}

template<typename S>
inline forme<S>::forme(point<S> pt)
{
    this->addPoint(pt);
}
```

```hpp
template<typename S>
inline vector<point<S>> forme<S>::getVecteurPoint()
{
    return this->vectorPt;
}

template<typename S>
void forme<S>::setVectorPoint(vector<point<S>> vector)
{
    this->vectorPt = vector;
}

template<typename S>
inline void forme<S>::addPoint(point<S> pt)
{
    this->vectorPt.push_back(pt);
}

template<typename S>
inline ostream& operator<<(ostream& s, forme<S> f)
{
    for (int i = 0; i < f.getVecteurPoint().size(); i++)
    {
        s << "pt "<< i <<" : " << f.getVecteurPoint()[i] << "\n";
    }
    return s;
}
```

PARTIE 3 : rectangle.h

```hpp
#pragma once
#include "forme.h"
template<typename R>
class rectangle : public forme<R>
{
protected :
    R hauteur;
    R largeur;
public:
    rectangle();
    rectangle(point<R> pt);
    R perimetre();
    R surface();
    R getLargeur();
    R getHauteur();

    void setLargeur(R l);
    void setHauteur(R h);
};

template<typename R>
inline rectangle<R>::rectangle()
{
}

template<typename R>
inline rectangle<R>::rectangle(point<R> pt):forme<R>(pt)
{
}

template<typename R>
inline R rectangle<R>::perimetre()
{
    return 2 * this->hauteur + 2 * this->largeur;
}

template<typename R>
inline R rectangle<R>::surface()
{
    return this->hauteur + this->largeur;
}

template<typename R>
inline R rectangle<R>::getLargeur()
{
    return this->largeur;
}

```

```hpp
template<typename R>
inline R rectangle<R>::getHauteur()
{
    return this->hauteur;
}

template<typename R>
inline void rectangle<R>::setLargeur(R l)
{
    this->largeur = l;
}

template<typename R>
inline void rectangle<R>::setHauteur(R h)
{
    this->hauteur = h;
}

template<typename R>
inline ostream& operator<<(ostream& s, rectangle<R> r)
{
    for (int i = 0; i < r.getVecteurPoint().size(); i++)
    {
        s << "pt " << i << " : " << r.getVecteurPoint()[i] << "\n";
    }
    s << "largeur " << r.getLargeur() << "\n";
    s << "hauteur " << r.getHauteur() << "\n";
    return s;
}

```